22:33:01 OCA PAD AMENDMENT - PROJECT HEADER INFORMATION 01/28/92

Active

Project #: E-25-649 Cost share #: E-25-353 Rev #: 2
Center #.: 10/24-6-R6808-0A0 Center shr #: 10/22-1-F6808-0A0 OCA file #:
Work type : RES
Contract#: MSS-8910427 Mod #: OPAS-NCE Document : GRANT
Prime #: Contract entity: GTRC

Subprojects ? : N CFDA: 47.041
Main project #: PE #: N/A

Project unit: MECH ENGR Unit code: 02.010.126
Project director(s):
SADEGH N MECH ENGR (404)894-8172

Sponsor/division names: NATL SCIENCE FOUNDATION / GENERAL
Sponsor/division codes: 107 / 000

Award period: 890801 to 920731 (performance) 921031 (reports)

Sponsor amount	New this change	Total to date
Contract value	0.00	70,000.00
Funded	0.00	70,000.00
Cost sharing amount		16,000.00

Does subcontracting plan apply ?: N

Title: LEARNING CONTROL OF MECHANICAL SYSTEMS

PROJECT ADMINISTRATION DATA

OCA contact: Mildred S. Heyser 894-4820

Sponsor technical contact

ELBERT MARSH
(202)357-9542

NSF
1800 G ST., N.W.
WASHINGTON, D.C. 20550

Sponsor issuing office

MATTHEW SMILAK
(202)357-9626

NATIONAL SCIENCE FOUNDATION
1800 G STREET, NW
WASHINGTON, DC 20550

Security class (U,C,S,TS) : U ONR resident rep. is ACO (Y/N): N
Defense priority rating : N/A NSF supplemental sheet
Equipment title vests with: Sponsor GIT X

Administrative comments -
TO EXTEND PERFORMANCE ENDING DATE TO JULY 31, 1992 VIA OPAS .

GEORGIA INSTITUTE OF TECHNOLOGY
OFFICE OF CONTRACT ADMINISTRATION

NOTICE OF PROJECT CLOSEOUT

Closeout Notice Date 11/09/92

Project No. E-25-649 Center No. 10/24-6-R6808-0A0

Project Director SADEGH N School/Lab MECH ENGR

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Contract/Grant No. MSS-8910427 Contract Entity GTRC

Prime Contract No.

Title LEARNING CONTROL OF MECHANICAL SYSTEMS

Effective Completion Date 920731 (Performance) 921031 (Reports)

Closeout Actions Required:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	N	
Final Report of Inventions and/or Subcontracts	N	
Government Property Inventory & Related Certificate	N	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

CommentsNSF LETTER OF CREDIT APPLIES.

Subproject Under Main Project No.

Continues Project No.

Distribution Required:

Project Director	Y
Administrative Network Representative	Y
GTRI Accounting/Grants and Contracts	Y
Procurement/Supply Services	Y
Research Property Managment	Y
Research Security Services	N
Reports Coordinator (OCA)	Y
GTRC	Y
Project File	Y
Other HARRY VANN FMD	Y
FRED CAIN OOD	Y

E-25-649

NATIONAL SCIENCE FOUNDATION
ANNUAL PROGRESS REPORT

PI Name and Address:
Nader Sadegh, Assistant Professor
George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology
Atlanta, Ga. 30332

Part I - PROJECT IDENTIFICATION INFORMATION

Program Official: Dr. Elbert L. Marsh
Program Name: Dynamic Systems and Control
Award dates: From: 8/89 To: 1/92
Institution: Georgia Institute of Technology
Award Number: MSS-8910427
Project Title: **Learning Control of Mechanical Systems**

II. Summary of the Completed Work:

The primary objective of this research was to design, analyze and implement a class of proposed learning controllers for non-linear mechanical systems. The two main proposed tasks of the project, which included the theoretical development and experimental evaluation of the proposed controller, has now been completed. The proposed learning controller does not require an exact knowledge of the system dynamics and is computationally efficient. The stability of the overall control system taking into account the full nonlinear dynamics of the system has also been proven. To investigate its practical feasibility, the control scheme was simulated using a realistic model of a Scara type robot. The simulation results demonstrated a significant improvement in performance compared to conventional PID schemes. A research article [1,2] based on this work was prepared and has been accepted for publication.

An IBM Scara robot (7545) robot has been modified to be used as a test-bed for the developed control schemes. The results obtained so far indicates a substantial (about 2 orders of magnitude) increase in tracking accuracy of the robot over the IBM's original controller. [3]

Other work resulting from this research includes some new theoretical results on repetitive control of more general dynamic systems. This work has been submitted for publication [4].

Currently under investigations are: (1) Implementation of the proposed controller in Cartesian space, (2) Neural-Network Learning Control design, (3) End-point sensing and control of flexible robots.

III. Technical Information

As a result of this award the following research articles have been submitted/and or accepted for publication in refereed journals or conference proceedings:

[1] N. Sadegh and K. Guglielmo, "A new Learning Controller for Mechanical Manipulators", Accepted for publication in the Journal of Robotic Systems, August, 1991.

[2] K. Guglielmo and N. Sadegh, "A new Learning Controller for Mechanical Manipulators Applied in Cartesian Space", To be presented at the ASME Winter Annual Meeting, Dallas, Texas, Nov. 1990.

[3] N. Sadegh, "Synthesis and Analysis of Repetitive Control Systems", Submitted to IEEE Trans. on Auto. Control for publication.

[4] K. Guglielmo and N. Sadegh, "Experimental Evaluation of a New Robot Learning Controller", Submitted to IEEE Int. Conference on Robotic and Automation, April, 1991.

IV. Abstracts of the Resulting Research Articles

See the attached sheets.

A New Learning Controller For Mechanical Manipulators

Nader Sadegh
Kennon Guglielmo

The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology, Atlanta Georgia 30318

ABSTRACT

A new learning controller for motion control of mechanical manipulators undergoing periodic tasks is developed. This controller does not require exact knowledge of the manipulator dynamic structure or its parameters, and is computationally efficient. In addition, no actual joint accelerations or any matrix inversions are needed in the control law. The global asymptotic stability of the ideal and the robust stability of the nonideal control system is proven taking into account the full nonlinear dynamics of the manipulator. Simulation results of this algorithm applied to a realistic Scara type manipulator [7], which includes dry friction, pay-load inertia variations, actuator/sensor noise, and unmodelled dynamics are also presented.

A New Learning Controller for Mechanical Manipulators Applied in Cartesian Space

Kennon Guglielmo
Nader Sadegh

Abstract

This paper presents a new learning controller for motion control of mechanical manipulators undergoing periodic tasks defined in Cartesian space. The controller does not require knowledge of the manipulator dynamic parameters beyond a simple geometric description. Feedback will be accomplished through the use of a Cartesian space position sensor for the end–effector, and standard joint position and joint velocity sensors. No Cartesian velocity signal and no acceleration feedback of any kind is required, and the Cartesian end– effector position may be easily generated using *forward* kinematics if direct measurement is not possible. The desired task will be defined in Cartesian coordinates, and no inverse kinematics or inverse Jacobian will be calculated. The asymptotic stability of this algorithm is proven using the Lyapunov approach, and the non–linear characteristics of the manipulator are explicitly taken into account.

Simulation results applied to a realistic SCARA type manipulator model are presented [11]. This model has been patterned very accurately after an actual manipulator and includes realistic friction, sensor/actuator noise, payload variation, high order dynamics, digital quantization, and actuator saturation effects. These simulation results confirm the asymptotic stability of the learning control scheme and also demonstrate robustness to simulation and trajectory induced disturbances.

1 Introduction

Most industrial robots working in assembly line applications perform a desired task repetitively with each task requiring a prescribed period of time. Just as humans become more proficient at performing a series of motions with practice, the "repetitive" or "learning" class of controllers enable robots to increase their tracking accuracy while performing a periodic task. As the manipulator works, position, velocity, and/or acceleration feedback is used to form error signals, and the "learning" law attempts to reduce these errors from one cycle to the next by modifying the calculated input torque.

Early works on learning or repetitive control can be found in [2,9,4,22]. The most common technique used in these works is a "delayed integral" action of the form $(1-e^{-sT})^{-1}$. Theoretically, this algorithm would learn the time history of the periodic disturbances and eventually cancel their effects after several cycles of the trajectory. In these works as well as more recent ones, such as

Synthesis and Stability Analysis of Repetitive Control Systems *

Nader Sadegh

The George W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology, Atlanta Georgia 30318

Abstract

A class of repetitive controllers for tracking control of dynamical systems subject to repetitive desired trajectories is presented. Within the framework of periodic functions, the necessary and sufficient conditions along with an explicit procedure for stably inverting a linear time–invariant plant is formulated. This procedure, which is also applicable to nonminimum phase plants, can be used in determining the *exact* feedforward input of the plant. Next, the stability of a repetitive control system, which is a distributed parameter system, is analyzed and the necessary and sufficient conditions for its *asymptotic* convergence are postulated. An extended Nyquist criteria for the stability analysis and synthesis of this controller is subsequently proposed. Finally, a modified repetitive controller is introduced, which expands the domain of applicability of the original controller to a wider class of plants, including nonminimum phase systems. The main stability results pertaining to the original controller are then extended to those of the modified controller. Several simulation examples and design guidelines are also presented. Simulation results confirm a perfect asymptotic tracking performance for the repetitive control systems that satisfy the stability conditions.

*This work was supported by the National Science Foundation under grant MSS-8910427.

Experimental Evaluation of a New Robot Learning Controller

Kennon Guglielmo
Nader Sadegh

The George W. Woodruff School of Mechanical Engineering
The Georgia Institute of Technology
Atlanta, Georgia, 30332-0405

Abstract

The results of the implementation of a new learning control algorithm on an IBM 7545 robotic manipulator are presented. Simulation studies of this repetitive controller have been presented in [8] and [17]. The implementation of the joint space algorithm on the IBM manipulator confirms and even exceeds the performance estimated by the previous simulation work. Feedback was obtained from optical joint position encoders, and velocity was estimated by simple numerical differentiation in software. No acceleration feedback of any kind was used, and no dynamic parameters, dynamic equations of motion, or kinematic equations were needed. The performance of the algorithm was compared to that of a simple PD feedback system (common in many current industrial robots), and an adaptive algorithm which performs at least as well as a "Computed Torque" controller. The learning algorithm outperformed both of these controllers by a huge margin, and exhibited convergence within approximately three cycles.

OMB Number 345-0058

NATIONAL SCIENCE FOUNDATION
1800 G STREET, NW
WASHINGTON, DC 20550

E25-619

BULK RATE
POSTAGE & FEES PAID
National Science Foundation
Permit No. G-69

PI/PD Name and Address

Nader Sadegh
Georgia Tech Research Foundation
Mechanical Engineering
Atlanta, GA 30332

NATIONAL SCIENCE FOUNDATION
FINAL PROJECT REPORT

PART I - PROJECT IDENTIFICATION INFORMATION

1. Program Official/Org. Devendra Garg

2. Program Name Dynamics Systems and Control

3. Award Dates (MM/YY) From: 8/1/89 To: 1/31/92 extended to 7/31/92

4. Institution and Address

Georgia Tech Research Foundation
Mechanical Engineering
Atlanta, GA 30332

5. Award Number 8910427

6. Project Title

Research Initiation Award: Learning Control of Mechanical Systems

This Packet Contains
NSF Form 98A
And 1 Return Envelope

NSF Grant Conditions (Article 17, GC-1, and Article 9, FDP-II) require submission of a Final Project Report (NSF Form 98A) to the NSF program officer no later than 90 days after the expiration of the award. Final Project Reports for expired awards must be received before new awards can be made (NSF Grant Policy Manual Section 677).

Below, or on a separate page attached to this form, provide a summary of the completed project and technical information. Be sure to include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

The summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the project title, it should begin with a topic sentence stating the project's major thesis. The summary should include, if pertinent to the project being described, the following items:

- **The primary objectives and scope of the project**
- **The techniques or approaches used only to the degree necessary for comprehension**
- **The findings and implications stated as concisely and informatively as possible**

PART III - TECHNICAL INFORMATION (for program management use)

List references to publications resulting from this award and briefly describe primary data, samples, physical collections, inventions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available to the research community. Provide the NSF Invention Disclosure number for any invention.

	10/30/92
Principal Investigator/Project Director Signature	**Date**

IMPORTANT:
MAILING INSTRUCTIONS
Return this *entire* packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the *Attention block* on the envelope.

PART IV — FINAL PROJECT REPORT — SUMMARY DATA ON PROJECT PERSONNEL

(To be submitted to cognizant Program Officer upon completion of project)

The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in response to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the information.

Please enter the numbers of individuals supported under this grant.
Do not enter information for individuals working less than 40 hours in any calendar year.

	Senior Staff		Post-Doctorals		Graduate Students		Under-Graduates		Other Participants[1]	
	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.
A. Total, U.S. Citizens					1					
B. Total, Permanent Residents	1									
U.S. Citizens or Permanent Residents[2]:										
American Indian or Alaskan Native ...										
Asian										
Black, Not of Hispanic Origin										
Hispanic										
Pacific Islander										
White, Not of Hispanic Origin	1				1					
C. Total, Other Non-U.S. Citizens										
Specify Country 1.										
2.										
3.										
D. Total, All participants (A + B + C)	1				1					
Disabled[3]										

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

[1]Category includes, for example, college and precollege teachers, conference and workshop participants.

[2]Use the category that best describes the ethnic/racial status for all U.S. Citizens and Non-citizens with Permanent Residency. *(If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)*

[3]A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. *(Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")*

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America, and who maintain cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia and the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific Territories of Guam, American Samoa, or the Northern Marianas; the U.S. Trust Territory of Palau; the islands of Micronesia or Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

Learning Control of Mechanical Systems

Prepared for the National Science Foundation

Principal Investigator:
Nader Sadegh

George W. Woodruff School of Mechanical Engineering
School of Mechanical Engineering

Part II–Summary of the Completed Work

The primary objective of this research was to design, analyze and implement a class of learning/repetitive controllers for non-linear mechanical systems. The main tasks of the project, which included theoretical development and experimental evaluation of the proposed controllers, have been successfully completed. The developed learning controllers require no exact knowledge of the system's model and are computationally efficient. They can be used as a "plug–in" module to significantly improve the tracking performance of an existing servo controller without significantly increasing its cost or complexity.

An IBM Scara (7545) robot was modified to serve as a test–bed for the implementation of the proposed control schemes. The implementation results demonstrated the feasibility of the learning controllers for applications in which high–performance position or force control would be required. These results also confirmed and even exceeded the performance estimated by the previously published simulation results [1,2]. The learning algorithms were compared to that of a simple PD feedback system (common in many current industrial robots), and a "Computed Torque" controller. They outperformed both of these controllers by a significant margin, and exhibited fast convergence.

Other work resulting from this project includes some new theoretical results on repetitive control of more general class of dynamical systems, and synthesis of neural network based learning controllers for nonlinear systems.

Part III–Technical Information

III-A Background

The increase in complexity of today's engineering systems, coupled with tighter requirements on their control performance, have necessitated a search for alternatives to the conventional control strategies. The conventional linear control techniques are often satisfactory for regulating a single plant with known dynamics about a certain operating point. As the operating range of the plant is expanded, more uncertainty is introduced to the control system and the nonlinear effects become more prominent. As a result, the performance of a conventional controller based on the linear model may deviate significantly from the desired behavior. The learning control techniques have been proposed in recent years as a promising approach to maintain a high level of performance in the presence of plant and environmental uncertainties, and to increase the autonomy of the control system.

In this research we investigate a class of learning controllers for mechanical systems. The mechanical systems considered here possess highly nonlinear dynamic equations of motion and are influenced by other nonlinear effects such as static friction and digital quantization. Typical "adaptive" control techniques attempt to use the *specific* structure of some model of the overall system and estimate unknown constant parameters within that model. The problem with this approach is that the model may not be known or may be incomplete. "Learning" control algorithms, such as those developed as a part of this project, avoid the use of specific modeling equations and attempt to estimate the overall input–output relationship of a system.

The general topologies of the controllers investigated in this project are similar in that they are all derived from a modified "Computed Torque" scheme (see [14, 15]). In this scheme, the nonlinear dynamics are "canceled" by feeding forward an input signal which is derived directly from the equations of motion of the manipulator. Some type of feedback loop is also present to add robustness to the system.

The computed torque method and its derivatives are considered superior to many other algorithms because the nonlinearities of the system are dealt with in a direct manner as opposed to some type of linearization scheme or the pure application of a linear feedback controller. This type of control is not without its problems and limitations, however. There is always some uncertainty associated with the dynamic inertial parameters included in the equations of motion used to form the feedforward signal. Also, some unmodeled effects are almost always present such as unknown friction mechanisms or unknown payloads. Adaptive control based on the computed torque method has been developed to deal with these problems. In many of these adaptive schemes (see [14] for examples) the dynamic equations of the manipulator are decomposed into a known nonlinear function of the state of the manipulator and an unknown vector of inertial and kinematic parameters. An update law driven by some type of error signal is then used to "adapt" the unknown parameter vector – hopefully to the true parameters of the robot.

An extension of adaptive controllers to include estimation of the entire feedforward term – not just unknown constants – has been given the name "learning" control. Many of these schemes require that the task being performed is periodic. This means that the manipulator is required to execute some trajectory with a finite period in a cyclic fashion. In this way

the class of "learning" or "repetitive" controllers acquires a knowledge of the feedforward quantity necessary to operate the robot with desirable performance. One problem with the "repetitive" class of learning controllers is that the desired task must be periodic. By exploring the general structure of the dynamic effects present in a robotic manipulator, a learning algorithm may be derived which relaxes the restriction of periodicity of the task. This new controller, referred to as the *Configuration Learning Law* (CLL), learns the functional relationship between the feedforward control input and the desired states of the system. As a result the desired task does no longer have to be periodic.

Up to this point we have only mentioned adaptive and learning *position* control of mechanical systems. A field even more recent than this is the adaptive or learning *force* control. In most of the existing works in this area, MRAC (Model Reference Adaptive Control) or simple gain tuning is used to maintain a desired contact force with a surface while moving tangentially to it with some prescribed motion. Few new control algorithm applications are explored and the performance of these controllers is questionable since simulation results are usually presented. Simulation studies are not very reliable since real surface interactions are often very difficult to model and are usually highly nonlinear. As a part of this project, we have developed and experimentally verified a learning controller, referred to as the *Hybrid Learning Law* (HLL) [3], for applications requiring force control. This controller enables the robot to "learn" how to accurately follow a desired path on an *unknown* surface while maintaining a prespecified contact force.

The algorithms developed in this research extend the application of learning control to several specific areas. This work should facilitate the use of learning control in the flexible manufacturing arena and provide a basis for application of these schemes to other linear and nonlinear systems.

III-B Completed Tasks of Project

We now briefly describe the completed phases of this project and the resulting publications.

Repetitive Controller

The goals of this phase of the project, which included the theoretical design and analysis of the proposed repetitive controller, have now been fulfilled. The objective of this controller, which serves as the backbone for the subsequent phases of this project, is to "learn" and generate the feedforward input torque for tracking a trajectory specified in either the joint or Cartesian space. When the task is specified in the Cartesian space, the controller is formulated so that it requires no kinematic inversions. Only forward kinematics, if direct measurement is not possible, is needed.

The way that this controller "learns" the required feedforward input can be described as follows: the periodic feedforward input is expressed as a linear combination of a countable set of appropriately selected periodic functions with *unknown* coefficients (e.g., Fourier series approximation). A learning control law, which uses a steepest descent parameter estimation law is employed to *directly* estimate a finite number of these unknown coefficients and use them in the calculation of the feedforward torque input. The stability and robustness of this controller to effects such as truncation errors and input disturbances was rigorously

analyzed. All of these analyses were carried out by considering the full nonlinear dynamic model of the manipulator.

The main advantages of this controller as compared to those in the previous works may be summarized as follows: (1) No need for joint acceleration measurements or any matrix inversions; (2) Reduction in memory space requirement since for most trajectories, a moderate number of coefficients suffices for approximation purposes; (3) Faster convergence rate and improved robustness to unmodeled disturbances, and (4) *parallel* processability of the learning algorithm. The results of this phase of the project have appeared in references [10, 11, 14].

Experimental Evaluation of Proposed Controllers

In this phase of the project, the control algorithms described previously were implemented on an IBM 7545 four degree of freedom robot using an Intel 80386/387 based microcomputer and associated hardware. The links of this robot are actuated by DC motors coupled to harmonic gear drives. The harmonic drives of the actuators introduce a significant degree of flexible unmodeled dynamics into the control system. In addition, a substantial amount of Coulomb friction is present at each joint. In spite of these undesirable effects, the implementation of the proposed algorithms confirmed and even exceeded the performance estimated by the earlier simulation results. It was demonstrated that the implemented repetitive control scheme not only outperformed the existing robot PID controller by a significant margin — about 2 orders of magnitudes in terms of the position tracking error — but also exhibited a faster convergence rate than that of the existing repetitive algorithms. In fact, the controller forced the robot to track the entire desired trajectory within the resolution capabilities of the position encoders after about four cycles. The main results of this work has been presented in [2, 7].

Repetitive Force Control

The previous position controller was extended to a hybrid learning force/position control scheme for tracking an unknown surface with a specified contact force. The algorithm used for all elements of the control scheme was a repetitive learning similar to that described above. The learning law was also applied to on–line trajectory generation for maintaining normal contact to the unknown surface. The complete control scheme was implemented on an IBM 7545 robot, and excellent performance was observed in terms of position, force, and orientation tracking. A Simple PID version of the overall control strategy was implemented for comparison purposes, and the learning controller outperformed the PID implementation by a large margin in all areas. The learning algorithm proved robust to non–periodic disturbances, and was almost as computationally efficient as the simple PID scheme. The preliminary results of this work have appeared in reference [3].

Configuration Learning Law

One limitation that all previously mentioned controllers have in common is that they all require a periodic task. Once a particular task is learned it is performed with little error. However, if the task is slightly altered, the entire learning process must be reactivated for

the new task. The Configuration Learning Law (CLL) has been developed to specifically address this problem. The CLL learns and generates the feedforward control input as a function of the desired states of the system. As a result the desired task does no longer have to be periodic. The preliminary experimental results of this work are reported in [1].

Repetitive Control of Nonminimum Phase Systems

One key property of a rigid robot manipulator that makes it attractive for control purposes is that it is a passive system from an input–output point of view. This passivity property is lost however for flexible robots with noncollocated input–outputs. Instability may occur if the aforementioned repetitive controller is applied to such a system. As part of this project, a modified repetitive controller was developed, which expanded the domain of applicability of the original controller to a wider class of plants, including the nonpassive and nonminimum phase systems.

Moreover, in the case of linear plants, a set of necessary and sufficient stability conditions was derived, which can be used for design and analysis of the repetitive controller. The results of this work have been appeared in references [9, 12].

Neural Network Based Control

The Artificial Neural Networks (ANN) are massively parallel computational machines capable of learning and reconstructing nonlinear mappings. These capabilities of the ANN have spurred an interest among the control researchers to apply them for solving complex control problems, especially those dealing with nonlinear systems. The main advantage of using the ANN as a controller is twofold: 1) complex nonlinear, even "table–look–up", control algorithms can be dynamically mapped onto the ANN, and recalled instantly when demanded; 2) the learning capability of the ANN enables the resulting controller to adapt itself to possible variations in the plant (i.e., system under control) dynamics while in operation.

Motivated by our previous work in the robotics area, we have introduced an ANN–based control methodology that can be used for a large class of nonlinear dynamical systems, which includes the previously mentioned mechanical systems as a subset. The structure of the control law incorporating the ANN network can be derived based on the identified and analytical nominal model of the nonlinear system using the geometrical control techniques for nonlinear dynamical systems. The performance of the proposed ANN controller has been investigated in several case studies including guidance of ground vehicles, and control of free–flying (nonholonomic) robots. These systems are chosen as their dynamic models are highly nonlinear and subject to uncertainties thus providing a suitable test–bed for the proposed control system.

The developed control scheme, as in the case of the robot controller, includes a feedback and a feedforward control component. An important feature that distinguishes this scheme from other existing neural network controllers is that the feedforward input is learned directly, without having first to identify the inverse dynamics of the plant. In most of the existing neural network controllers this process is carried out indirectly — first, an network is trained to model the *inverse* plant dynamics, and then the trained network is used to compute the control input. The main advantages of the direct controller are that 1) it can react to unexpected changes and disturbances more effectively, 2) it has a faster convergence

rate, 3) in general, it requires fewer processing elements, and 4) it requires no inversion of the plant dynamics. The main results of this research have appeared in references [8, 6, 13].

References

[1] Guglielmo, K. and N. Sadegh, "A Non–Repetitive Learning Controller for Mechanical Manipulators," *Proceedings of the 1992 ASME Winter Annual Meeting,* Anaheim, California, November 8–13, 1992.

[2] Guglielmo, K. and N. Sadegh, "Motion Control of Mechanical Manipulators," Journal of Intelligent and Robotic Systems, vol. 6, June 1992, pp. 17–31.

[3] Guglielmo, K. and N. Sadegh, "Topology and Implementation of a learning Hybrid Force Control Scheme," *Proceedings of the 1992 Japan–USA Symposium on Flexible Automation,* San Francisco, California, July, 1992, pp. 843–848.

[4] Guglielmo, K. and N. Sadegh, "A New Adaptive Controller with Generalized Friction Estimation," *Proceedings of the 1991 ASME Winter Annual Meeting,* Atlanta, Georgia, November 1991.

[5] Guglielmo, K. and N. Sadegh, "A New Learning Controller for Mechanical Manipulators Applied in Cartesian Space," *Proceedings of the 1990 ASME Winter Annual Meeting,* Dallas, Texas, 1990.

[6] Sadegh, N., "Learning Kinematic Control of a Redundant Manipulator," *Proceedings of the 1992 ASME Winter Annual Meeting,* Anaheim, California, November 8–13, 1992.

[7] Sadegh, N. and K. Guglielmo, "Design and Implementation of Adaptive and Repetitive Controllers for Mechanical Manipulators," *IEEE Transactions on Robotics and Automation,* vol. 8, no. 3, June 1992, pp. 395–400.

[8] Sadegh, N., "A Perceptron Network for Functional Identification and Nonlinear Control," *Proceedings of the 1992 American Control Conference,* Chicago, Illinois, June 1992. Also, Submitted to the *IEEE Transactions on Neural Networks.*

[9] Sadegh, N., "A Discrete–Time MIMO Repetitive Controller," *Proceedings of the 1992 American Control Conference,* Chicago, Illinois, June 1992.

[10] Sadegh, N., "Robustness Techniques in Nonlinear Systems with Applications to Manipulator Adaptive Control," *Control and Dynamic Systems:Advances in Theory and Applications,* Edited by C.T. Leondes, vol. 50, Academic Press, 1992.

[11] Sadegh, N. and K. Guglielmo, "A New Repetitive Controller for Mechanical Manipulators," the *Journal of Robotic Systems*, vol. 8(4), August 1991, pp. 507–529.

[12] Sadegh, N.,"Synthesis and Stability Analysis of Repetitive Control Systems," *Proceedings of the 1991 American Control Conference,* Boston, Massachusetts, June 24–25, 1991. Also, Submitted to IEEE Transactions on Auto. Contr. for Publication.

[13] Sadegh, N., "Nonlinear Identification and Control via Neural Networks," *Proceedings of the 1991 ASME Winter Annual Meeting,* Atlanta, Georgia, November 1991.

[14] Sadegh, N. and R. Horowitz, "Stability and Robustness Analysis of a Class of Adaptive Controllers for Robotic Manipulators," *International Journal of Robotics Research*, Vol. 9, No. 3, June 1990, pp. 74–92.

[15] Sadegh, N., R. Horowitz, W.-W. Kao, and M. Tomizuka, "A Unified Approach to the Design of Adaptive and Repetitive Controllers for Robotic Manipulators," *Journal of Dynamic Systems, Measurement, and Control,* vol. 112, pp. 618–629, December 1990.

[16] Sadegh, N. and R. Horowitz, "Stability Analysis of an Adaptive Controller for Robotic Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 1223–1239, Raleigh, North Carolina, March 1987.

III-C Ph.D. Dissertation Resulting form this Award

The remainder of this report contains the complete Ph.D. Dissertation of Kennon Guglielmo who was partially supported under this award. This dissertation includes the theory and implementation details of the learning controllers developed for nonlinear mechanical systems.

Learning Position and Force Control for Mechanical Manipulators

A DISSERTATION
Presented to
The Academic Faculty

by

Kennon Guglielmo

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Mechanical Engineering

Georgia Institute of Technology
May 4, 1992

Acknowledgments

First and foremost I would like to thank my wife, Laura. Without her support and encouragement I would have been many more months completing this dissertation, and without her my life would be very dreary and empty.

I would also like to thank my mother for the wonderful education she has provided me, and for the wonderful life she has given me. Her example and beliefs were the basis for my relationship with God whose strength and forgiveness allow me to live my life and accomplish those things that I seek.

On the technical side, I would like to thank Dr. Nader Sadegh for all of his understanding, creativity, insight, and support in helping me complete my research and this dissertation. His suggestions and viewpoints have been invaluable.

I would also like to thank Dr. Wayne Book and Dr. Y. H. Chen for their advice and instruction in the classroom, as well as their patience and time in sitting on my thesis committee.

Also, I would like to thank Dr. Bonnie Heck and Dr. David Taylor for taking time out of their electrical engineering duties to serve on my committee, and for their constructive questions and suggestions concerning my research.

Last but not least, I would like to thank Dr. John Peatman for the opportunity and the means to expand my electrical engineering skills. His friendship and encouragement have been invaluable to my growth as a well rounded mechanical engineer.

Contents

List of Figures

Summary

This dissertation addresses the application of "learning" control to robotic manipulators. Learning control attempts to estimate the feedforward input necessary to force the robot to perform some desired task with zero error. This feedforward signal is estimated in a general way so that very little modeling or identification of the system is involved. The task may be position control described in either the joint or Cartesian space of the manipulator, or force control as the robot interacts with an object. Two basic forms of the learning control law are developed. This first is a "repetitive" controller which requires the desired trajectory to be cyclic with a finite period. The second is a "non–repetitive" scheme which relaxes the restriction of periodicity of the task, but requires some very general modeling of the system. Stability of the learning algorithms is proven using the Lyapunov approach, and all of the controllers are implemented on an actual IBM 7545 robot. Performance comparisons to simple classical feedback algorithms as well as some high performance "computed–torque" and adaptive controllers show that the learning schemes are capable of obtaining significantly better tracking in all cases.

All of the new learning controllers presented are computationally and memory efficient, and are easily implemented on today's digital hardware. In fact, one of the "non–repetitive" schemes can be viewed as an efficient form of a "neural network" capable of being implemented in real–time on a low cost microcomputer. Although a robotic manipulator was chosen as the test bed for implementation studies of these learning controllers, many other nonlinear systems may benefit from application of these algorithms. In fact, the extension of the basic topology of the learning scheme to these different areas of robotic control demonstrates the algorithm's easy applicability. The work presented in formulating the controllers in terms of the nonlinear equations of motion of a mechanical manipulator provides an outline for extension to other systems.

Chapter 1

Introduction

The robotic systems considered in this dissertation possess highly nonlinear dynamic equations of motion and are influenced by other nonlinear effects such as static friction and digital quantization. Typical "adaptive" control techniques attempt to use the *specific* structure of some model of the overall system and estimate unknown constant parameters within that model. The problem with this approach is that the model may not be known or may be incomplete. "Learning" control algorithms, such as those developed in this dissertation, avoid the use of specific modeling equations and attempt to estimate the overall input–output relationship of a system.

The general topologies of the controllers investigated in this dissertation are similar in that they are all derived from a modified "Computed Torque" scheme (see [10,3,30]). In this scheme, the nonlinear dynamics are "canceled" by feeding forward an input signal which is obtained directly from the equations of motion of the manipulator. Some type of feedback loop is also present to add robustness to the system. A controller of this type referred to as the *Desired Compensation Control Law* (DCCL) has been presented in [30] and will be used here for comparison purposes to the new learning algorithms.

The computed torque method and its derivatives are considered superior to many other algorithms because the nonlinearities of the system are dealt with in a direct manner as opposed to some type of linearization scheme or the pure application of a linear feedback

controller. This type of control is not without its problems and limitations, however. There is always some uncertainty associated with the dynamic inertial parameters included in the equations of motion used to form the feedforward signal. Also, some unmodelled effects are almost always present such as unknown friction mechanisms or unknown payloads. Adaptive control based on the computed torque method has been developed to deal with these problems.

In many of these adaptive schemes (see [11,30,31,35] for examples) the dynamic equations of the manipulator are decomposed into a known nonlinear function of the state of the manipulator and an unknown vector of inertial and kinematic parameters. An update law driven by some type of error signal is then used to "adapt" the unknown parameter vector – hopefully to the true parameters of the robot. The general concept of an update law can be understood using the following static example.

Given some function $\mathbf{Y}$ which can be expressed as the product of a time varying matrix $\mathbf{W}(t)$ and a constant vector $\mathbf{\Theta}$:

$$\mathbf{Y} = \mathbf{W}(t)\mathbf{\Theta} \tag{1.1}$$

an estimate of $\mathbf{Y}$ is given by:

$$\hat{\mathbf{Y}} = \mathbf{W}(t)\hat{\mathbf{\Theta}} \tag{1.2}$$

where $\hat{\mathbf{\Theta}}$ is an estimate of $\mathbf{\Theta}$. The update law for $\hat{\mathbf{\Theta}}$ is given by:

$$\dot{\hat{\mathbf{\Theta}}} = -\mathbf{K}\mathbf{W}^T(t)\mathbf{e} \tag{1.3}$$

where $\mathbf{K}$ is a positive definite gain matrix and $\mathbf{e}$ is the function estimate error defined by:

$$\mathbf{e} = \hat{\mathbf{Y}} - \mathbf{Y} \tag{1.4}$$

In this way we force the parameter estimate to the true parameter vector so that the function estimate $\hat{\mathbf{Y}}$ follows $\mathbf{Y}$. This update law has come to be known as the "steepest descent" law in the controls literature, and it is important since all of the learning algorithms developed in this dissertation use it in some form. By using this law to update the parameters

within the DCCL's feedforward compensation, we can derive an adaptive form of this controller. This algorithm, referred to as the *Desired Compensation Adaptive Law* (DCAL), has been presented in [30] and will be used here for comparison purposes to the new learning algorithms.

An extension of adaptive controllers to include estimation of the entire feedforward term – not just unknown constants – has been given the name "learning" control. Many of these schemes require that the task being performed is periodic. This means that the manipulator is required to execute some trajectory with a finite period T in a cyclic fashion. In this way the class of "learning" or "repetitive" controllers acquires a knowledge of the feedforward quantity necessary to operate the robot with desirable performance. Many of the early works on learning controllers can be found in [2,17,4,36,20,5]. In these works as well as many others, one or more of the following assumptions or requirements are usually needed to accomplish the controller design and the convergence analysis:

1. The inverse of the manipulator inertia matrix must be calculated on–line.
2. The joint accelerations must be available.
3. The dynamics of the manipulator must be *approximated* by a linear time–varying differential equation.

In [32], a new repetitive learning control law is presented in which the above restrictions and assumptions are relaxed. The key ideas behind this approach which make the relaxations possible are:

1. The properties of the nonlinear dynamic structure of the manipulator are utilized by creating a Lyapunov function which resembles the total energy of the robot, and by using a feedback control law which makes this function non–increasing [30,34].
2. The feedforward input to the manipulator, which may include the inertial, Coriolis, gravity, and friction effects, is decomposed into two terms:
   (a) a nonlinear function of the position and velocity errors
   (b) a *periodic* term, which is only a function of the *desired* trajectory signals

A *delayed integral* type controller was used in this particular scheme to generate the periodic term. However, a controller of this type requires a huge memory space for digital

implementation because of its infinite dimensional dynamics. Also, simulation studies show that controllers of this type demonstrate a slow convergence rate when non–periodic and impulsive disturbances are present [37,32]. This scheme is important, however, since the repetitive learning algorithm presented in this dissertation has the same structure.

In [33], another new *joint space* repetitive controller referred to as the *Desired Compensation Learning Law* (DCLL) is developed. By joint space we mean that the trajectory description and feedback are in terms of the joint angles of the manipulator. By using a linear combination of unknown coefficients and known periodic "shape functions," the required feedforward compensation can be formed for a periodic task. The general form of the previously mentioned "steepest descent" update law for the unknown coefficients is used. In [16,14] a *Cartesian space* version of this same repetitive control law is presented in which the trajectory description and feedback are in terms of some Cartesian coordinate system. We will refer to this controller as the *Desired Compensation Cartesian Learning Law* (DCCLL).

Up to this point we have only mentioned adaptive and learning *position* control of manipulators. A field even more recent than this is the adaptive or learning *force* control of manipulators. Some recent works in this area can be found in [23,9,24,22]. In these papers and others, MRAC (Model Reference Adaptive Control) or simple gain tuning is used to maintain a desired contact force with a surface while moving tangentially to it with some prescribed motion. Few new control algorithm applications are explored and the performance of these controllers is questionable since simulation results are usually presented. Simulation studies are not very reliable since real surface interactions are often difficult to model and are usually highly nonlinear. A new repetitive learning algorithm specifically formulated for force control will be developed which is based on the DCCLL. Actually, this controller will be a combination of Cartesian position control, force control, and on–line trajectory update, and will be referred to as the *Hybrid Learning Law* (HLL).

One problem with the "repetitive" class of learning controllers is that the desired task must be periodic. By exploring the general structure of the dynamic effects present in a

robotic manipulator, a learning algorithm may be derived which relaxes the restriction of periodicity of the task. As previously mentioned, unknown friction mechanisms may exist in a robot which consume a large portion of the control effort. Accurate cancellation of these effects without requiring a specific model may be achieved with non–repetitive learning control. The *Friction Compensation Adaptive Law* (FCAL) implements this concept by using the basic structure of the DCAL along with a learning term which is a function of the joint *velocities.*

Although the FCAL does not require the desired trajectory to be periodic, the dynamic equations of motion of the system must be derived because of the inclusion of the DCAL portion of the feedforward signal. A new controller referred to as the *Configuration Learning Law* (CLL) addresses this problem by allowing the equations of motion to be expressed in only a general form. This scheme uses the same velocity dependent friction learning term as the FCAL as well as several *position* dependent terms to approximate the specific equation of motion for the system.

The algorithms developed in this dissertation extend the application of learning control to several specific areas. This work should facilitate the use of learning control in the flexible manufacturing arena and provide a basis for application of these schemes to other linear and nonlinear systems.

Chapter 2

IBM 7545 Manipulator

2.1 Introduction

In this chapter we will describe the hardware and software used for implementation studies of the controllers developed in this dissertation. Although computer simulations of dynamic systems offer some insight into the relative performance of a controller, accurate evaluation of an algorithm's computational efficiency, robustness qualities, and ultimate performance must be accomplished by implementation on a physical system. Toward this end, an IBM 7545 four degree of freedom (DOF) robot was modified for use as a controller test bed.

This robot was introduced in the United States in 1984, and is actually a modified Japanese design from Sankyo, Inc. The principal use of this manipulator was printed circuit board (PCB) "stuffing" (i.e. electronic part installation). Its four DOF design allowed enough mobility for this type of planar insertion task, and its SCARA configuration offered compliance in the arm so that misaligned parts could mate properly.

This manipulator's open kinematic chain design gives rise to some highly nonlinear dynamic effects which will be discussed in chapter 3. Since the robot is a nonlinear dynamic system, it is an excellent test bed for evaluation of nonlinear control algorithms. In the next section we will discuss the system configuration in terms of actuators, amplifiers, link design, and available feedback, and in the last section of this chapter we will detail modifications

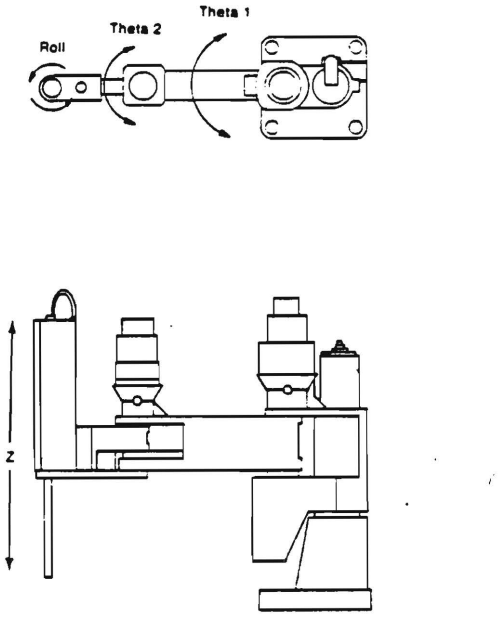

Figure 2.1: IBM 7545 Four Degree of Freedom Robot

of the manipulator for use as a generic controller test system.

2.2 System Configuration

As previously mentioned, the IBM 7545 manipulator used for controller performance comparison in this dissertation is a four DOF device using an open kinematic chain design. Its first two degrees of freedom consist of revolute joints with their axes aligned in a vertical direction. This is a typical SCARA, or specified compliance, configuration offering compliance in the horizontal plane based on the feedback system in place around the actuators. The third DOF is a prismatic "z" axis generating movement in the vertical direction, and the last DOF is a "roll" motion about the z axis which allows independent orientation in the horizontal plane.

The z axis is naturally dynamically decoupled from the horizontal plane motion generated by the action of the first two revolute joints. Due to the belt driven design of the "roll" axis, this DOF should also be decoupled from the other degrees of freedom. However, since friction is present in the bearings and pulleys used to transmit motion to this axis, some

coupling does occur.

All of the actuators are brush–type DC motors coupled to high reduction harmonic gear drives. The gear reduction allows the IBM 7545 to generate high accelerations, but limits the maximum link velocity due to speed limitations of the motors. The harmonic drives offer the advantage of zero backlash, but this comes at a cost of increased flexibility over a "standard" type gear reduction. The prismatic z axis is of the preloaded ball screw type. This design again offers very low backlash, but has the drawback of highly nonlinear and large friction effects.

The first two links are very rigid, and the flexibility of this portion of the manipulator is essentially due to the harmonic drives. The z axis has highly variable flexibility which is a function of its extension. When the z axis is fully retracted, the system is much stiffer than when this axis is extended. This is because the z axis is supported at both ends when retracted, but becomes effectively cantilever supported from one end when extended. All of the experiments in this dissertation were performed with the z axis at most only slightly extended to avoid large flexibility effects. Table 2.2 gives the link lengths and travel for each axis of the robot.

Feedback of the manipulator position is provided via optical incremental encoders located on the motors driving each link of the robot. Because these encoders are located on the motor side of the gear reduction system, their resolution is multiplied by the gear ratio. Table 2.3 gives the encoder resolution for each axis of the robot.

	Length	*−Travel*	*+Travel*
1^{st} *Axis*	400mm	−15°	+205°
2^{nd} *Axis*	250mm	−15°	+135°
Z Axis	–	0mm	+250mm
Roll Axis	–	−185°	+185°

Figure 2.2: IBM 7545 Link Specifications

The DC motors were driven by pulse width modulation (PWM) type power amplifiers.

	Resolution
1^{st} *Axis*	±0.0011°
2^{nd} *Axis*	±0.0023°
Z Axis	±0.0026mm
Roll Axis	±0.0044°

Figure 2.3: IBM 7545 Optical Encoder 4X Quadrature Resolutions

These amplifiers were originally designed and used in the manufacturer's control system as PI (Proportional–Integral) velocity control servos. A tachometer signal was derived from the optical encoders using a frequency to voltage converter, and was fed back to the amplifiers. These units have been modified for current command control by utilizing their current detection circuit for feedback instead of the tachometer signal. Since a DC motor's torque is proportional to the current flowing through it, these amplifiers provide an effective torque command capability. This modification was done to facilitate application of control algorithms formulated in terms of torque/force input to the actuators. The amplifier dynamics are very fast in comparison to the sampling rates used for the digital portion of the control schemes implemented in this dissertation. Although it may be conjectured that these dynamics have some slight effect on *this* robot's performance, investigation and cancellation of these fast effects is an area of control unto itself and will not be directly addressed.

2.3 Computer Integration

In this section we will discuss the problems and details associated with modification of the IBM 7545 robot for use as a controller test system. The original controller consisted of an inner analog PI velocity feedback loop surrounded by an outer digital proportional position feedback. The analog PI velocity servo was described in the previous section, and the digital position feedback loop was implemented using an on–board microprocessor. This microprocessor was not user programmable in terms of the controller software, and had

to be removed from the control loop. The analog PI velocity servo amplifiers were also modified for current (torque) demand as described previously.

A microcomputer based (PC) control platform was chosen to replace the original digital control system for several reasons:

1. Low cost
2. Computing power similar to that available for industrial systems
3. Standard bus for easy connection to interface cards
4. Excellent programming software already developed and available
5. Easy upgradability
6. Multipurpose system capable of running other software

Originally, an Intel 80386 25MHz based PC was chosen, but this computer has since been upgraded to an 80486 25MHz unit. Both of these computers offered more than enough computing power to implement the controllers presented in this dissertation at high sampling rates.

To obtain link position information from the robot, the optical encoder's quadrature signals were fed into a Keithley/Metrabtye, Inc., bus mounted encoder board. This board was capable of monitoring all four of the IBM 7545's encoders simultaneously, and came equipped with software drivers for accessing the position information. The modified analog power amplifiers were driven by a Keithley/Metrabyte digital to analog output board. Many of the digital signals such as proximity switch outputs and gripper actuation inputs were connected to the PC through digital I/O ports located on this same board.

An ATI, Inc., wrist force/torque sensor was also added to the robot to facilitate implementation of the force control algorithms to be presented in chapter 8. This sensor was equipped with an Intel 8255 based digital parallel interface which was connected to the PC via a Keithley/Metrabyte digital I/O board. Although the sensor also provided an RS232 based serial interface, the parallel system was used for higher speed feedback. This sensor provided the six Cartesian force and torque components felt by the manipulator at its end–effector at an update rate of 500Hz.

Controller code was principally written in Microsoft C with several assembly language subroutine components. The structure of the software provided a very simple means for implementing a control law either for actual control of the robot or simulation using a near exact model of the system. A "control module" was simply written in C to perform the required controller computations using input and output routines from a single library. This library contained all of the file manipulation, robot interface, kinematic calculation, and initialization routines necessary for controller support. For actual robot control, the control module was linked to the implementation library, and for simulation studies, the same control module was linked to the simulation library. From the control module's point of view, the library routines looked exactly alike for both implementation and simulation. The advantage of this system was that a new controller could be written, debugged, and tuned before ever having to perform a run on the actual robot. Once the new controller was working properly, it could simply be relinked with the implementation library and executed on the physical manipulator.

The typical digital feedback rate for all of the controllers evaluated in this dissertation was 500Hz. Many of the adaptive or learning portions of the algorithms were updated at 125Hz to demonstrate their parallel processing capability and their easy implementation on slower digital hardware. We may express this dual rate control by using the following digital equivalent of the continuous time controllers presented in later chapters:

$$\mathbf{q}(k) = \mathbf{q}_{fb}(k) + \mathbf{q}_{ff}(\overline{k}) \tag{2.1}$$

where $\mathbf{q}(k)$ is the input to the actuators, $\mathbf{q}_{fb}(k)$ is the input due to the *feedback* portion of the control law, and $\mathbf{q}_{ff}(\overline{k})$ is the input due to the *feedforward* portion. k is the index for the feedback portion of the controller, $\overline{k}$ is the index for the adaptive or learning portion, and m is an integer which relates these indices according to $k = m\overline{k}$. Each increment in k represents a time step of Δt seconds, and each increment in $\overline{k}$ represents a time step of $m\Delta t$ seconds. To implement the 500Hz feedback and 125Hz feedforward rates previously mentioned, $\Delta t = 2ms$ and $m = 4$.

All of these controllers could have been operated at 1500Hz on the original 80386 based PC, and 3000Hz on the newer 80486 PC. These sampling rates include not only the time for control law computation, but also error history storage and other non–necessary functions that would not be present in a real–time application of these algorithms.

In general, the IBM 7545 robot and associated hardware and software for controller implementation provides an excellent vehicle for control algorithm evaluation. Its dynamic equations of motion are inherently nonlinear, the harmonic gear reduction drives possess highly nonlinear and time varying friction characteristics, and digital quantization, input/output time delays, and input/output noise are present. All of these effects serve to accurately test the performance of a control algorithm on a real world system.

Gain selection for implementation of the controllers is a topic of interest. The feedback portion of the control law given in equation 2.1 consist of a PD (Proportional–Derivative) term in all of the control algorithms presented in this dissertation. These gains may be easily estimated by first linearizing the robot equations of motion about some nominal operating point, and then using standard classical control design techniques [8,10] for a linear PD controller. At this point the algorithm should achieve stable, although not necessarily high performance, trajectory tracking with the feedforward portion $\mathbf{q}_{ff}$ set to zero. With the feedback loop in operation, it is now a straightforward task to find any gains associated with the feedforward (learning) term. In each of the learning control schemes, the DCLL, FCAL, CLL, and HLL, only one learning gain is associated with each axis of the coordinate space. Therefore, each learning gain may be adjusted in a pseudo–decoupled fashion to achieve the desired speed of convergence and steady state error (see [16] for a detailed Cartesian gain selection description).

Chapter 3

Dynamics of Robotic Manipulators

3.1 Introduction

In this chapter we will review the dynamic equations of motion for an n dimensional manipulator. We define this manipulator to be a Lagrangian system with n degrees of freedom consisting of an open kinematic chain with n links. Also, each of its degrees of freedom will be equipped with an independent actuator. We will further assume that the number of degrees of freedom is less than or equal to 6 (i.e. a non–redundant robot). The controllers developed in subsequent chapters will either be formulated in the *joint* or *Cartesian* space of the manipulator. For this reason, the following two sections will address the equations of motion for the robot in terms of each coordinate system.

3.2 Joint Space Dynamics

In this section we will review the *joint* space dynamics of robotic manipulators. Let $\mathbf{M}_j(.)$ denote the joint space inertia matrix, which is bounded, positive definite and locally C^∞ in the position $\mathbf{x}_j$ [21]. Using the Lagrangian formulation in the joint space of the manipulator,

the equations of motion of an n degree of freedom robot may be expressed by:

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_j &= \dot{\mathbf{x}}_j \\ \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\dot{\mathbf{x}}_j + \mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{V}\dot{\mathbf{x}}_j &= \mathbf{q} + \mathbf{d} \end{aligned} \tag{3.1}$$

where $\mathbf{x}_j$ and $\dot{\mathbf{x}}_j$ are the position and velocity vectors, $\mathbf{g}(\mathbf{x}_j)$ represents the force due to gravity, $\mathbf{V}$ is the diagonal matrix of *viscous* joint friction coefficients, $\mathbf{q}$ is the input force supplied by the actuators to the manipulator joints, and $\mathbf{d}$ summarizes the effect of input disturbances such as unmodelled friction, actuator noise, and contact forces. The ij–th element of $\mathbf{C}$, c_{ij}, also known as the Christoffel symbol, is given by:

$$c_{ij} = \frac{1}{2}\sum_{k=1}^{n} \dot{x}_{pk}\left[\frac{\partial m_{ij}}{\partial x_{pk}} + \frac{\partial m_{ik}}{\partial x_{pj}} - \frac{\partial m_{jk}}{\partial x_{pi}}\right] \tag{3.2}$$

where m_{ij} is the ij–th element of $\mathbf{M}_j$, x_{pj} is the j–th element of $\mathbf{x}_j$ and $\dot{x}_{pk}$ is the k–th element of $\dot{\mathbf{x}}_j$ [32,30]. Also see [7] for a more general treatment. Note that the term $\mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j$ is usually referred to as the Coriolis vector in the robotics literature.

Example 3.2.1

This example specifically gives the joint space inertia matrix $\mathbf{M}_j(\mathbf{x}_j)$*, the Coriolis matrix* $\mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)$*, the gravity vector* $\mathbf{g}(\mathbf{x}_j)$*, and the viscous friction matrix* $\mathbf{V}$*, for the first two revolute links of the IBM 7545 manipulator in terms of the constant inertial parameters* Θ_1 *—* Θ_5*:*

$$\mathbf{M}_j(\mathbf{x}_j) = \begin{bmatrix} \Theta_1 + 2\Theta_3\cos(x_{j2}) & \Theta_2 + \Theta_3\cos(x_{j2}) \\ \Theta_2 + \Theta_3\cos(x_{j2}) & \Theta_2 \end{bmatrix} \tag{3.3}$$

$$\mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j) = \begin{bmatrix} -\Theta_3\dot{x}_{j2}\sin(x_{j2}) & -\Theta_3(\dot{x}_{j1}+\dot{x}_{j2})\sin(x_{j2}) \\ \Theta_3\dot{x}_{j1}\sin(x_{j2}) & 0 \end{bmatrix} \tag{3.4}$$

$$\mathbf{g}(\mathbf{x}_j) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \tag{3.5}$$

$$\mathbf{V} = \begin{bmatrix} \Theta_4 & 0 \\ 0 & \Theta_5 \end{bmatrix} \tag{3.6}$$

Note that the viscous friction matrix $\mathbf{V}$ *was included in this formulation due to the large amount of friction present in the IBM 7545 manipulator.*

The dynamics of a manipulator have some special properties which are central to the convergence proofs presented in this paper. Although these properties are well known [30], we present them here for completeness:

Definition 3.2.1 *The* covariant derivative *of a C^1 vector field* $\mathbf{v}(.) : \mathbf{R}^n \rightarrow \mathbf{R}^n$ along *a C^1 curve* $\mathbf{x}_p(.) : \mathbf{R}^+ \rightarrow \mathbf{R}^n$, *denoted with*

$$\frac{D}{dt} \text{ is defined by}$$

$$\frac{D}{dt}(\mathbf{v}(\mathbf{x}_p), \mathbf{x}_p) = \frac{d}{dt}\mathbf{v}(\mathbf{x}_p) + \mathbf{M}^{-1}(\mathbf{x}_p)\mathbf{C}(\mathbf{x}_p, \dot{\mathbf{x}}_p)\mathbf{v}(\mathbf{x}_p) \tag{3.7}$$

To show the covariant derivative has properties similar to those of an ordinary derivative, we present the following theorem:

Theorem 3.2.1 *Given two vector fields* $\mathbf{v}_1$ *and* $\mathbf{v}_2$ *in* $\mathbf{R}^n$ *and scalar functions* f_1 *and* f_2: $\mathbf{R}^n \rightarrow \mathbf{R}$, *the following relations hold:*

$$\frac{D_p}{dt}(f_1\mathbf{v}_1 + f_2\mathbf{v}_2) = f_1\frac{D_p}{dt}(\mathbf{v}_1) + f_2\frac{D_p}{dt}(\mathbf{v}_2) + \frac{d}{dt}f_1(\mathbf{x}_p)\mathbf{v}_1 + \frac{d}{dt}f_2(\mathbf{x}_p)\mathbf{v}_2 \tag{3.8}$$

and

$$\frac{d}{dt}(\mathbf{v}_1^T\mathbf{M}(\mathbf{x}_p)\mathbf{v}_2) = \mathbf{v}_2^T\mathbf{M}(\mathbf{x}_p)\frac{D_p}{dt}\mathbf{v}_1 + \mathbf{v}_1^T\mathbf{M}(\mathbf{x}_p)\frac{D_p}{dt}\mathbf{v}_2 \tag{3.9}$$

where we have used the notation

$$\frac{D_p}{dt}\mathbf{v} \equiv \frac{D}{dt}(\mathbf{v}(\mathbf{x}_p), \mathbf{x}_p)$$

Proof: The first property is an immediate consequence of the definition. The second property follows from the *skew symmetry* of the matrix $\left[\dot{\mathbf{M}}(\mathbf{x}_p) - 2\mathbf{C}(\mathbf{x}_p, \dot{\mathbf{x}}_p)\right]$ [35,30]. For a complete proof see [29]. □

By combining the result of equation 3.9 with equation 3.7, we obtain the following simple relationship:

$$\frac{d}{dt}\left(\mathbf{v}^T\mathbf{M}(\mathbf{x}_p)\mathbf{v}\right) = 2\mathbf{v}^T\left[\mathbf{M}(\mathbf{x}_p)\frac{d}{dt}\mathbf{v} + \mathbf{C}(\mathbf{x}_p, \dot{\mathbf{x}}_p)\mathbf{v}\right] \tag{3.10}$$

where $\mathbf{v}_1 = \mathbf{v}_2 = \mathbf{v}$. This equation will be used in the Lyapunov analysis of the following chapters.

3.3 Cartesian Space Dynamics

In this section we will review the *Cartesian* space dynamics of the robotic manipulator. Let $\mathbf{M}_c(.)$ denote the Cartesian inertia matrix which is *assumed* to be bounded, positive definite and locally C^∞ in the position $\mathbf{x}_c$ [21]. Note that the assumption of $\mathbf{M}_c$ being bounded is not valid when the manipulator is near a singular position. In other words, when the manipulator configuration places the end–effector at the work space boundary the inertia matrix defined for Cartesian generalized coordinates becomes infinite.

As an example, let us explore the relationship between the joint space inertia matrix $\mathbf{M}_j(\mathbf{x}_j)$ and the Cartesian space inertia matrix $\mathbf{M}_c(\mathbf{x}_c)$.

Definition 3.3.1 *If* $\mathbf{x}_c = \varsigma(\mathbf{x}_j)$ *then define:*

$$\mathbf{J}(\mathbf{x}_j) \equiv \frac{d\varsigma(\mathbf{x}_j)}{d\mathbf{x}_j} \tag{3.11}$$

This $\mathbf{J}$ is known in the robotic literature as the manipulator Jacobian. By using its transformation properties on the joint space inertia matrix, we obtain the following (see [29], [3] for details):

$$\mathbf{M}_c(\mathbf{x}_c) = \mathbf{J}^{-T}(\mathbf{x}_j)\mathbf{M}_j(\mathbf{x}_j)\mathbf{J}^{-1}(\mathbf{x}_j) \tag{3.12}$$

Since the Jacobian matrix becomes singular in the singular positions of the manipulator [3], the inertia matrix for the Cartesian coordinates, $\mathbf{M}_c(\mathbf{x}_c)$, becomes unbounded near singularities. This poses a problem for direct Cartesian space control. The convergence proofs for this type of algorithm and the controller itself break down if the desired trajectory passes through a singular position of the manipulator. We will require that any desired trajectory which the manipulator will perform never ventures closer than some δ to the workspace boundary. Therefore, we will make the following assumption:

Assumption: The desired *Cartesian* position trajectory is confined to a *connected and compact* set S in the *interior* of the manipulator workspace such that

$$\sigma_{\min}\left(\mathbf{J}(\mathbf{x}_d)\right) > \delta \qquad \forall\ \mathbf{x}_d \in S \tag{3.13}$$

where $\sigma_{\min}$ denotes the smallest singular value, and $\mathbf{x}_d$ is the desired position vector. In this way we avoid the effective infinite inertia phenomenon. If the manipulator *must* pass through or very near a singular position, then a joint space scheme may be "switched on" temporarily until the robot is clear of the workspace boundary.

The equation of motion in terms of the Cartesian space of the manipulator is written as:

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_c &= \dot{\mathbf{x}}_c \\ \mathbf{M}_c(\mathbf{x}_c)\frac{d}{dt}\dot{\mathbf{x}}_c + \mathbf{C}(\mathbf{x}_c, \dot{\mathbf{x}}_c)\dot{\mathbf{x}}_c + \mathbf{g}_c(\mathbf{x}_c) &= \mathbf{q} + \mathbf{d} \end{aligned} \tag{3.14}$$

where $\mathbf{x}_c$ and $\dot{\mathbf{x}}_c$ are the Cartesian position and velocity vectors, $\mathbf{g}_c(\mathbf{x}_c)$ represents the Cartesian force due to gravity, $\mathbf{q}$ is the input force supplied by the actuators expressed in terms of Cartesian coordinates at the end–effector, and $\mathbf{d}$ once again summarizes the effect of input disturbances such as friction, actuator noise, contact forces, etc. $\mathbf{C}(\mathbf{x}_c, \dot{\mathbf{x}}_c)$ may be obtained in an analogous fashion to the joint space quantity using equation 3.2.

Chapter 4

Adaptive and Non–Adaptive Control Laws

4.1 Introduction

In this chapter we will review the development of the *Desired Compensation Control Law* (DCCL) and the *Desired Compensation Adaptive Law* (DCAL) presented in [30]. These algorithms are applicable to joint space control of robotic manipulators performing either periodic or non–periodic tasks, and are included here for two reasons. The first is that they are closely related to the learning laws (both repetitive and non–repetitive) which will be presented in later chapters, and the second is that they will be used for performance comparison to these same learning algorithms. Some of the similarities which exist between the DCCL, the DCAL, and the learning laws are:

1. The same PD (Proportional Derivative) *feedback* loop, nonlinear error compensation, and *feedforward* structure is used.
2. The same *steepest descent* update law is used to estimate unknown parameters in the feedforward signal (DCAL and learning only).
3. The *desired* trajectory signals are used, as opposed to the actual ones, to generate the feedforward signal.
4. The same Lyapunov stability approach is used in the analysis.

The DCCL is actually a modified *computed torque* [10,3] non–adaptive algorithm which uses the system equations of motion in joint space with fixed dynamic parameters to form the feedforward signal. It will be presented first to preface the development of the DCAL, which is the adaptive version of this algorithm. Before a detailed description of each controller is presented, we will outline some basic definitions which will be used throughout this dissertation.

We will refer to the *desired* trajectory of the manipulator as $\mathbf{x}_d(t)$:

$$\mathbf{x}_d(t) = \text{desired position} \tag{4.1}$$

$$\dot{\mathbf{x}}_d(t) = \frac{d}{dt}\mathbf{x}_d(t) = \text{desired velocity} \tag{4.2}$$

$$\ddot{\mathbf{x}}_d(t) = \frac{d}{dt}\dot{\mathbf{x}}_d(t) = \text{desired acceleration} \tag{4.3}$$

We will restrict the set of *allowable* desired trajectories for the controllers presented in this and following chapters to be "bounded" in the following sense:

Definition 4.1.1 *First, let C^2 be the space of twice continuously differentiable functions. A subset of C^2, denoted by A_d, is said to be a set of allowable desired trajectories if*

$$\sup_{\mathbf{f}\in A_d}\sup_{t\geq 0}\left|\frac{d}{dt}\mathbf{f}(t)\right| < \infty \quad \text{and} \quad \sup_{\mathbf{f}\in A_d}\sup_{t\geq 0}\left|\frac{d^2}{dt^2}\mathbf{f}(t)\right| < \infty \tag{4.4}$$

This simply means that we require the first and second derivatives of the desired trajectory, $\dot{\mathbf{x}}_d(t)$ and $\ddot{\mathbf{x}}_d(t)$, to be bounded. By assuming that every $\mathbf{x}_d(t)$ we choose belongs to a particular A_d, the subsequent stability results will hold for *any* allowable desired trajectory.

4.2 Desired Compensation Control Law

The DCCL, as well as most of the other controllers presented in this dissertation, consist of three main parts:

1. A linear PD feedback compensation.
2. A nonlinear feedback compensation.
3. A feedforward compensation.

The PD control action, which uses an inner velocity and an outer position loop, together with the nonlinear portion of the control law are used to guarantee the exponential stability of the overall system. It should be noted that both of these portions of the control law have *fixed* gains. The feedforward term, which is a function of the *desired* trajectory signals and manipulator dynamic parameters, is used to provide the required feedforward joint force/torque input for trajectory following purposes.

To formulate the feedback controller, we first need to define the trajectory following position and velocity errors. Define the actual joint position and velocity vectors as $\mathbf{x}_j(t)$ and $\dot{\mathbf{x}}_j(t)$, and define the joint tracking position error, $\mathbf{e}(t)$, by:

$$\mathbf{e} = \mathbf{x}_j - \mathbf{x}_d \tag{4.5}$$

In order to guarantee the exponential stability of the algorithm, we need to introduce an auxiliary signal, $\mathbf{v}_j(t)$, which will be the reference velocity input to the inner velocity loop.

$$\mathbf{v}_j = \dot{\mathbf{x}}_d - \lambda \mathbf{e} \qquad \lambda > 0 \tag{4.6}$$

We will define the error corresponding to the reference and actual velocity as the *reference velocity error*, $\mathbf{e}_v(t)$:

$$\mathbf{e}_v = \dot{\mathbf{x}}_j - \mathbf{v}_j = \dot{\mathbf{e}} + \lambda \mathbf{e} \tag{4.7}$$

The control law, which determines the joint force/torque, is given by:

$$\mathbf{q} = -\mathbf{K}_p \mathbf{e} - \mathbf{K}_v \mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_d \tag{4.8}$$

$\mathbf{K}_p$ and $\mathbf{K}_v$ are the positive definite PD gain matrices. $\mathbf{q}_n$ is the nonlinear feedback term given by:

$$\mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) = \sigma_n |\mathbf{e}|^2 \mathbf{e}_v \qquad \sigma_n > 0 \tag{4.9}$$

and $\hat{\mathbf{w}}_d$ is an estimate of the feedforward compensation term whose desired value is given by:

$$\mathbf{w}_d = \mathbf{M}_j(\mathbf{x}_d)\ddot{\mathbf{x}}_d + \mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \mathbf{V}\dot{\mathbf{x}}_d \tag{4.10}$$

Remark: If the desired trajectory quantities, $\mathbf{x}_d$, $\dot{\mathbf{x}}_d$, $\ddot{\mathbf{x}}_d$, are known in advance, the computation of the $\mathbf{w}_d(t)$ vector, needed for the feedforward compensation, can be performed off–line.

Remark: The viscous friction term, $\mathbf{V}\dot{\mathbf{x}}_d$, was specifically included in the formulation of $\mathbf{w}_d$ because of the presence of high viscous friction in the harmonic drives of the IBM 7545 manipulator.

Example 4.2.1

Note that equation 4.10 can be reparameterized by separating the functional portion of the equation of motion from the constant dynamic parameters in the following way:

$$\mathbf{w}_d = \mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\boldsymbol{\Theta} \tag{4.11}$$

where $\boldsymbol{\Theta}$ is the vector of constant dynamic parameters. The specific form of $\mathbf{W}$ for the first two axes of the IBM 7545 robot is:

$$\mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d) =$$
$$\begin{bmatrix} \ddot{x}_{d1} & \ddot{x}_{d2} & (2\ddot{x}_{d1} + \ddot{x}_{d2})\cos(x_{d2}) - (2\dot{x}_{d1}\dot{x}_{d2} + \dot{x}_{d2}^2)\sin(x_{d2}) & \dot{x}_{d1} & 0 \\ 0 & \ddot{x}_{d1} + \ddot{x}_{d2} & \ddot{x}_{d1}\cos(x_{d2}) + \dot{x}_{d1}^2\sin(x_{d2}) & 0 & \dot{x}_{d2} \end{bmatrix}$$

and the actual value of the constant parameter vector, $\boldsymbol{\Theta}$, is:

$$\boldsymbol{\Theta} = \begin{bmatrix} 0.550 & 0.035 & 0.045 & 1.3 & 0.2 \end{bmatrix}^T$$

Applying the control law given by equation 4.8 to the manipulator system whose dynamics are governed by equation 3.1, we obtain the following error dynamics:

$$\begin{aligned} \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n - \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) + \overline{\mathbf{d}} \end{aligned} \tag{4.12}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda\mathbf{e} \tag{4.13}$$

where

$$\overline{\mathbf{d}} = \mathbf{d} + \hat{\mathbf{w}}_d - \mathbf{w}_d \tag{4.14}$$

and is defined as the *feedforward estimation error*. $\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e})$ is the resulting disturbance due to the use of the *desired* trajectory signals, instead of the *actual* ones, in the feedforward compensation and is defined by:

$$\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) = \left[\mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{v}_j + \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{v}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{V}\dot{\mathbf{x}}_j\right] - \mathbf{w}_d \tag{4.15}$$

We need the following lemma to establish the bounds on $\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e})$.

Lemma 4.2.1 *For the error system in equation 4.12, the following equation holds:*

$$\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) = -\lambda\mathbf{M}_j(\mathbf{x}_j)\mathbf{e}_v + \lambda^2\mathbf{M}_j(\mathbf{x}_j)\mathbf{e} + \Delta'\mathbf{w}(\mathbf{e}_v, \mathbf{e}) \tag{4.16}$$

where

$$|\Delta'\mathbf{w}(\mathbf{e}_v, \mathbf{e})| \leq b_1 |\mathbf{e}_v| + b_2 |\mathbf{e}| + b_3 \left[|\mathbf{e}_v| |\mathbf{e}| + \lambda |\mathbf{e}|^2\right] \tag{4.17}$$

where b_1, b_2, and b_3 are all positive valued functions, bounded by $|\dot{\mathbf{x}}_d|$ and $|\ddot{\mathbf{x}}_d|$.

Proof: See [30] and [15,16].

We now present the following stability theorem for the DCCL:

Theorem 4.2.1 *For a set of allowable desired trajectories, A_d, the disturbance free error system described by equations 4.12 (i.e. $\overline{\mathbf{d}} = 0$), which results from the application of the control law given by equation 4.8 to the manipulator system governed by equation 3.1 is* globally exponentially stable, *i.e. both $\mathbf{e}_v(t)$ and $\mathbf{e}(t)$ converge to zero exponentially from a given initial condition, provided that the control gains $\mathbf{K}_p$, $\mathbf{K}_v$, λ, and σ_n are chosen sufficiently large.*

Proof: The proof of this theorem is based on the Lyapunov approach. Choose the following Lyapunov function candidate:

$$V(t, \mathbf{e}_v, \mathbf{e}) = \frac{1}{2}\mathbf{e}_v{}^T\mathbf{M}_j(\mathbf{e} + \mathbf{x}_d)\mathbf{e}_v + \frac{1}{2}\mathbf{e}^T\mathbf{K}_p\mathbf{e} \tag{4.18}$$

Note that V is both decrescent and locally positive definite. Differentiating $V(t, \mathbf{e}_v, \mathbf{e})$ with respect to t and using equation 3.9, we obtain:

$$\frac{d}{dt}V = \mathbf{e}_v{}^T\left[\mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v + \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{e}_v\right] + \mathbf{e}^T\mathbf{K}_p\frac{d}{dt}\mathbf{e} \tag{4.19}$$

Upon using the error equations 4.12 we have:

$$\frac{d}{dt}V = -\mathbf{e_v}^T\mathbf{K_v}\mathbf{e_v} - \lambda\mathbf{e}^T\mathbf{K_p}\mathbf{e} - \mathbf{e_v}^T\Delta\mathbf{w}(\mathbf{e_v},\mathbf{e}) - \mathbf{e_v}^T\mathbf{q_n}(\mathbf{e_v},\mathbf{e}) \tag{4.20}$$

After invoking the result of lemma 4.2.1 and performing some tedious but straightforward algebra (see [32] for details), we obtain:

$$\frac{d}{dt}V \leq -\mathbf{e_v}^T\overline{\mathbf{K}}_v\mathbf{e_v} - \lambda\mathbf{e}^T\overline{\mathbf{K}}_p\mathbf{e} \tag{4.21}$$

where $\mathbf{0} < \overline{\mathbf{K}}_v \leq \mathbf{K_v}$ and $\mathbf{0} < \overline{\mathbf{K}}_p \leq \mathbf{K}_p$. Thus the system is uniformly stable. We can also write

$$\frac{d}{dt}V \leq -\overline{\sigma}_v\mathbf{e_v}^T\mathbf{M}_j(\mathbf{x}_j)\mathbf{e_v} - \lambda\mathbf{e}^T\overline{\mathbf{K}}_p\mathbf{e} \tag{4.22}$$

$$\overline{\sigma}_v = \sigma_{\min}\left(\mathbf{M}_j^{-T/2}\overline{\mathbf{K}}_v\mathbf{M}_j^{-1/2}\right) > 0$$

where $\sigma_{\min}$ denotes the smallest singular value. Consequently,

$$\frac{d}{dt}V \leq -\gamma V \leq 0 \text{ where } \gamma = \min(\overline{\sigma}_v, \lambda) > 0 \tag{4.23}$$

Integrating equation 4.23 we have:

$$V(t,\mathbf{e_v},\mathbf{e}) \leq V(0,\mathbf{e}_{v0},\mathbf{e}_0)\, e^{-\gamma T} \tag{4.24}$$

Thus, $V(t,\mathbf{e_v},\mathbf{e})$ and consequently both $\mathbf{e_v}(t)$ and $\mathbf{e}(t)$ converge to zero exponentially. □

Corollary 4.2.1 *Under the hypothesis of theorem 4.2.1, the perturbed system described by equations 4.12 (i.e. $\overline{\mathbf{d}} \neq 0$) is $\mathbf{L}_\infty$ input/output stable, in the sense that there exist positive constants, α_1, α_2, β_1, and β_2 such that:*

$$\|\mathbf{e_v}(.)\|_\infty \leq \alpha_1\|\overline{\mathbf{d}}(.)\|_\infty + \alpha_2 \tag{4.25}$$

$$\|\mathbf{e}(.)\|_\infty \leq \beta_1\|\overline{\mathbf{d}}(.)\|_\infty + \beta_2 \tag{4.26}$$

Remark: This corollary states that if our estimation of the manipulator dynamics is not exact, the norm of the resulting error will be proportional to the parameter error norm and the norm of the other possible disturbances.

Proof: See [30] and [16]. □

This concludes the section on the DCCL. The development of the algorithm and the stability proof were presented here in detail so that the controllers described in later chapters may be presented in more brevity.

4.3 Desired Compensation Adaptive Law

This is the adaptive version of the DCCL joint space controller described in the previous section. The DCAL uses the same structure as the DCCL, except that the dynamic parameter vector is updated on–line as the controller operates. For a detailed development of the DCAL see [30].

The control law for the DCAL, which determines the joint force/torque, is given by:

$$\mathbf{q} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_v\mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_d \tag{4.27}$$

where $\mathbf{K}_p$ and $\mathbf{K}_v$ are again the positive definite PD gain matrices, and $\mathbf{q}_n$ is the nonlinear feedback defined in equation 4.9. $\hat{\mathbf{w}}_d$ is an estimate of the feedforward compensation similar to that of the DCCL whose desired value is given by:

$$\mathbf{w}_d = \mathbf{M}(\mathbf{x}_d)\ddot{\mathbf{x}}_d + \mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \mathbf{V}\dot{\mathbf{x}}_d \tag{4.28}$$

Using the reparameterization of equation 4.11 in example 4.2.1, we may also write

$$\mathbf{w}_d = \mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\boldsymbol{\Theta} \tag{4.29}$$

where $\boldsymbol{\Theta}$ is the vector of constant dynamic parameters. As in the DCCL, $\mathbf{W}(t)$ may be calculated off–line if the desired trajectory quantities are known in advance.

An estimate of $\mathbf{w}_d$ is given by

$$\hat{\mathbf{w}}_d = \mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\hat{\boldsymbol{\Theta}} \tag{4.30}$$

where $\hat{\Theta}$ is an estimate of the parameter vector Θ. The following standard parameter update law with constant adaptation gains is used:

$$\frac{d}{dt}\hat{\Theta} = -\mathbf{K}_a \mathbf{W}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_v \qquad \mathbf{K}_a > 0 \tag{4.31}$$

where $\mathbf{K}_a$ is the positive definite adaptation gain matrix and the parameters $\hat{\Theta}$ are updated on–line using a simple numerical linear integration routine.

After applying the control law of equation 4.27 and the parameter estimation law of equation 4.31 to the manipulator system given by 3.1, we arrive at the following error dynamic system:

$$\begin{aligned} \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n - \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) + \mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\tilde{\Theta} + \mathbf{d} \end{aligned} \tag{4.32}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda\mathbf{e} \tag{4.33}$$

$$\frac{d}{dt}\tilde{\Theta} = -\mathbf{K}_a \mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)^T\mathbf{e}_v \tag{4.34}$$

where $\tilde{\Theta}$ represents the *parameter estimation error* given by

$$\tilde{\Theta} = \hat{\Theta} - \Theta \tag{4.35}$$

and $\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e})$ is the same as that given in 4.15. The following lemma will be used to linearize this term about the desired trajectory in the subsequent stability analysis.

Lemma 4.3.1 *The term* $\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e})$ *given by equation 4.15 is continuously differentiable with respect to* $\mathbf{e}$ *and* $\mathbf{e}_v$, *and* $\Delta\mathbf{w}(0,0) = \mathbf{0}$ *for all* $t \geq 0$. *Moreover, the partial derivatives*

$$\begin{aligned} \Delta_1 &\equiv \frac{\partial \Delta\mathbf{w}}{\partial \mathbf{e}}(\mathbf{0}, \mathbf{0}) \\ \Delta_2 &\equiv \frac{\partial \Delta\mathbf{w}}{\partial \mathbf{e}_v}(\mathbf{0}, \mathbf{0}) \end{aligned}$$

are uniformly bounded such that $\|\Delta_1\| < \overline{\Delta}_1$ *and* $\|\Delta_2\| < \overline{\Delta}_2$.

Proof: It is easily seen from equation 4.15 that $\Delta\mathbf{w}(\mathbf{0}, \mathbf{0}) = \mathbf{0}$ and that $\Delta\mathbf{w}$ is C^∞ with respect to $\mathbf{e}_v$ and $\mathbf{e}$. Therefore, since the desired trajectory signals are bounded, the partial derivatives evaluated along this trajectory are bounded. □

We will also make use of the following definition to ensure convergence of the parameter vector $\hat{\Theta}$:

Definition 4.3.1 *The "weighting" matrix* $\mathbf{W}(t)$ *is said to be* persistently exciting *if there exist positive scalars* γ_1, γ_2, *and* s *such that*

$$\gamma_1 \mathbf{I} \geq \int_t^{t+s} \mathbf{W}^T(\tau)\mathbf{W}(\tau)d\tau \geq \gamma_2 \mathbf{I} \tag{4.36}$$

for all $t > 0$.

The following theorems address the global and local stability of the DCAL. These theorems are general in that they will be referenced by the other adaptive (learning) controllers to be presented in subsequent chapters.

Theorem 4.3.1 *There exists a set of control gains* $\mathbf{K}_p$, $\mathbf{K}_v$, *and* σ_n *such that the ideal (i.e.* $\mathbf{d} = \mathbf{0}$*) manipulator error system given by equations 4.32–4.33 is* globally asymptotically stable.

Proof: We prove the theorem by constructing a suitable Lyapunov function similar to the one used in the proof of theorem 4.2.1:

$$V(t) = \frac{1}{2}\mathbf{e}_v{}^T\mathbf{M}_j(\mathbf{e}+\mathbf{x}_d)\mathbf{e}_v + \frac{1}{2}\mathbf{e}^T\mathbf{K}_p\mathbf{e} + \frac{1}{2}\tilde{\Theta}^T\mathbf{K}_a^{-1}\tilde{\Theta} \tag{4.37}$$

Note that the function $V(t)$ is both decrescent and positive definite hence it is a Lyapunov function candidate [38]. Differentiating equation 4.37 with respect to time, and following steps which parallel the proof of theorem 4.2.1, we obtain:

$$\frac{d}{dt}V(t) \leq -\mathbf{e}_v{}^T\overline{\mathbf{K}}_v\mathbf{e}_v - \lambda\mathbf{e}^T\overline{\mathbf{K}}_p\mathbf{e} \tag{4.38}$$

where $0 < \overline{\mathbf{K}}_v \leq \mathbf{K}_v$ and $0 < \overline{\mathbf{K}}_p \leq \mathbf{K}_p$ are the same as those introduced in theorem 4.2.1. Therefore, $\frac{d}{dt}V(t) \leq 0$, and the system is globally asymptotically stable, i.e. $\mathbf{e}$ and $\mathbf{e}_v$ converge to zero asymptotically starting from any initial condition. □

Theorem 4.3.2 *Select* $\mathbf{z} = [\mathbf{e}_v{}^T\ \mathbf{e}^T\ \tilde{\Theta}^T]^T$ *as the state vector. Then, if* $\mathbf{W}(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)$ *is persistently exciting, there exists a set of control gains* $\mathbf{K}_p$ *and* $\mathbf{K}_v$ *such that the ideal error system of theorem 4.3.1 is* locally exponentially stable - *i.e. there exist an* $h > 0$, $\alpha > 0$ *and* $M > 0$ *such that if* $\mathbf{z}(t_0) \in B_h$, *then*

$$|\mathbf{z}(t)| \leq Me^{-\alpha(t-t_0)}|\mathbf{z}(t_0)| \quad \forall \quad t_0 \geq 0$$

where B_h *denotes a ball of radius* h *centered at the origin.*

Proof: We prove this theorem by the *indirect* Lyapunov method. Consider the original nonlinear error system of equations 4.32–4.34 expressed in the standard form

$$\frac{d}{dt}\mathbf{z} = \mathbf{f}(t, \mathbf{z}) \tag{4.39}$$

where

$$\mathbf{f}(t,\mathbf{z}) = \begin{bmatrix} \mathbf{F}(t,\mathbf{z})\left[-\mathbf{G}(t,\mathbf{z}) - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n - \Delta\mathbf{w}(\mathbf{e}_v,\mathbf{e}) + \mathbf{W}(\mathbf{x}_d,\dot{\mathbf{x}}_d,\ddot{\mathbf{x}}_d)\tilde{\Theta}\right] \\ -\lambda\mathbf{e} + \mathbf{e}_v \\ -\mathbf{K}_a\mathbf{W}(\mathbf{x}_d,\dot{\mathbf{x}}_d,\ddot{\mathbf{x}}_d)^T\mathbf{e}_v \end{bmatrix} \tag{4.40}$$

and

$$\mathbf{F}(t,\mathbf{z}) = \mathbf{M}_j^{-1}(\mathbf{x}_d + \mathbf{e}) \tag{4.41}$$

$$\mathbf{G}(t,\mathbf{z}) = \mathbf{C}(\mathbf{e} + \mathbf{x}_d, \mathbf{e}_v - \lambda\mathbf{e} + \dot{\mathbf{x}}_d) + \mathbf{K}_v \tag{4.42}$$

It is easily seen that $\mathbf{f}(t,\mathbf{z})$ is C^∞ — hence *locally* Lipschitz — in $\mathbf{z}$.

Linearizing this system about the equilibrium state $\mathbf{z} = \mathbf{0}$, we obtain the following set of linear time varying differential equations:

$$\frac{d}{dt}\mathbf{z} = \mathbf{A}(t)\mathbf{z} \tag{4.43}$$

where

$$\mathbf{A}(t) = \begin{bmatrix} -\mathbf{M}_j^{-1}(\mathbf{x}_d)[\mathbf{K}_v + \mathbf{C}(\mathbf{x}_d,\dot{\mathbf{x}}_d) + \Delta_1] & -\mathbf{M}_j^{-1}(\mathbf{x}_d)[\mathbf{K}_p + \Delta_2] & -\mathbf{M}_j^{-1}(\mathbf{x}_d)\mathbf{W} \\ \mathbf{I} & -\lambda\mathbf{I} & \mathbf{0} \\ -\mathbf{K}_a\mathbf{W}^T & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{4.44}$$

Following exactly the same method of analysis as in theorem 4.3.1, and invoking the result from [30] for the convergence of the parameter error vector $\tilde{\mathbf{\Theta}}$ to zero under the persistent excitation condition on $\mathbf{W}(t)$, we can prove that the system $\dot{\mathbf{z}} = \mathbf{A}(t)\mathbf{z}$ is uniformly asymptotically stable. However, exponential stability and uniform asymptotic stability are equivalent for linear systems. Now by combining the indirect Lyapunov theorem ([38], p. 188, theorem 5.4.21) and the converse of the Lyapunov theorem ([6], p. 28, theorem 1.5.1), it follows that 4.39 and therefore the original nonlinear error system is *locally exponentially stable* in the sense of the statement of the theorem. □

In light of the exponential stability results obtained in theorem 4.3.2, we may now show that the overall control system is *robust* to disturbances and unmodelled dynamics.

Theorem 4.3.3 *Consider the original manipulator error system given by equations 4.32–4.34. Under the hypothesis of theorem 4.3.2, there exist* γ, c *and* $h > 0$ *such that if* $\|\mathbf{d}\|_\infty \leq c$ *and* $\mathbf{z}(0) \in B_h$, *then*

1. $\mathbf{z}(t)$ *remains bounded for all time.*
2. $\mathbf{z}(t)$ *converges to a* B_δ *ball of radius* $\delta = \gamma\|\mathbf{d}\|_\infty$, *or more precisely,* $\overline{\lim}_{t\to\infty}|\mathbf{z}(t)| \leq \delta$ *where* $\overline{\lim}$ *denotes the* upper *limit.*

Proof: Similar to the previous theorem, equations 4.32–4.34 can be expressed in the form

$$\frac{d}{dt}\mathbf{z} = \mathbf{f}(t,\mathbf{z}) + \mathbf{F}(t,\mathbf{z})\mathbf{d} \tag{4.45}$$

where $\mathbf{f}$ and $\mathbf{F}$ are the same as the ones defined in 4.40. In addition to the properties stated in the proof of theorem 4.3.2 for $\mathbf{f}$ as a function of t and $\mathbf{z}$, it is easily seen that the right side of 4.45 is also Lipschitz in $\mathbf{d}$. Thus the differential 4.45 satisfies the hypothesis of the Small Signal I/O Stability theorem in ([6] p. 221, theorem 5.3.1) and the conclusions of the theorem follow. □

This concludes the development of the non–adaptive and adaptive controllers which will be used for performance comparison to the learning controllers in later chapters. Many of the definitions and terms presented in this chapter will be referred to in later chapters in

the interest of brevity. In the following section we present some performance comparisons for these controllers.

4.4 DCCL and DCAL Implementation Results

Both the DCCL and DCAL were implemented on the IBM 7545 manipulator to compare their performance to each other and to a simple PD feedback law. These algorithms were applied to joint space control as described in the previous sections. The joint space trajectory is shown in figure 4.1.

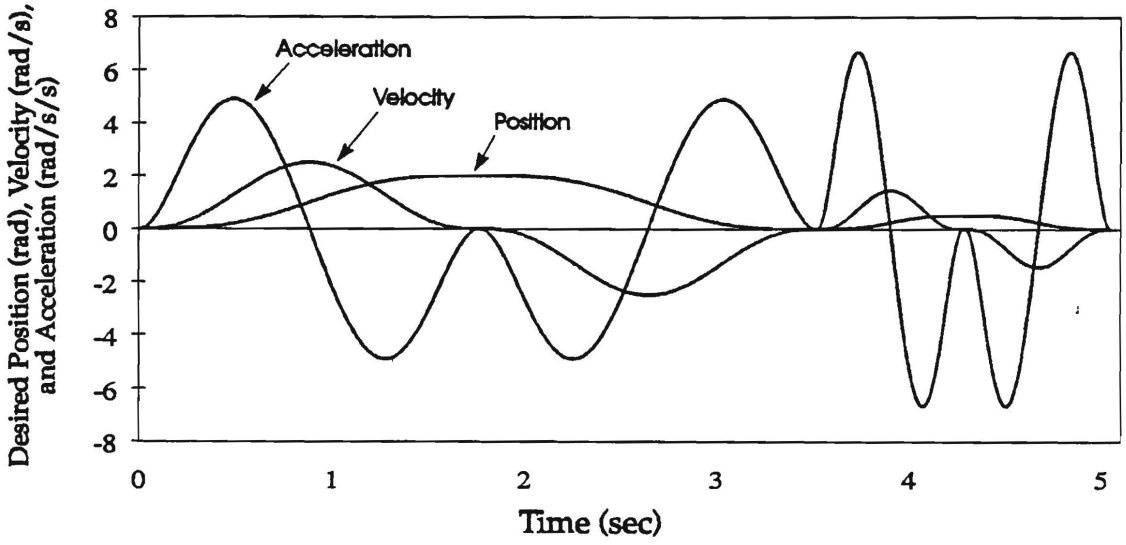

Figure 4.1: Joint Space Test Trajectory

This trajectory was prepared using a 7^{th} order polynomial for the desired position, and the velocities and accelerations shown correspond to the maximums specified for the DC motors and power amplifiers. Only the performance of the first two joints of the IBM 7545 will be presented in this and other implementation results sections because, as mentioned in chapter 2, these are the only links possessing nonlinear and coupled dynamic equations of motion.

Figures 4.2, 4.3, and 4.4 show the position error time history for the PD only, DCCL,

and DCAL control laws over six cycles of the desired trajectory. Note that all of these plots are on the same ordinate scale, and that the difference in performance is entirely due to the *feedforward* signal since all of the controllers use identical PD *feedback* loops. The DCAL's dynamic parameters quickly converge to appropriate values within the first few seconds of the trajectory. Steady state performance of the DCAL is better than that for the DCCL because the DCAL is able to "reshape" its feedforward signal on–line to achieve better tracking. This action demonstrates that there are effects in the system which are not represented by the model used for the DCCL and DCAL's feedforward signals.

Figures 4.5 and 4.6 show the RMS (root mean square) position error of all three controllers for the first and sixth cycles of the desired trajectory. These figures represent an average of the RMS error for the first two axes. The DCAL "steady state" position error betters the DCCL by 53%, and the PD feedback by 91%. The DCAL's ability to absorb some unmodelled disturbances by perturbing the dynamic parameters demonstrates the need for a more flexible way of forming the feedforward signal. Since the DCAL possesses the best "steady state" performance, it will be used for comparison to learning schemes developed in later chapters.

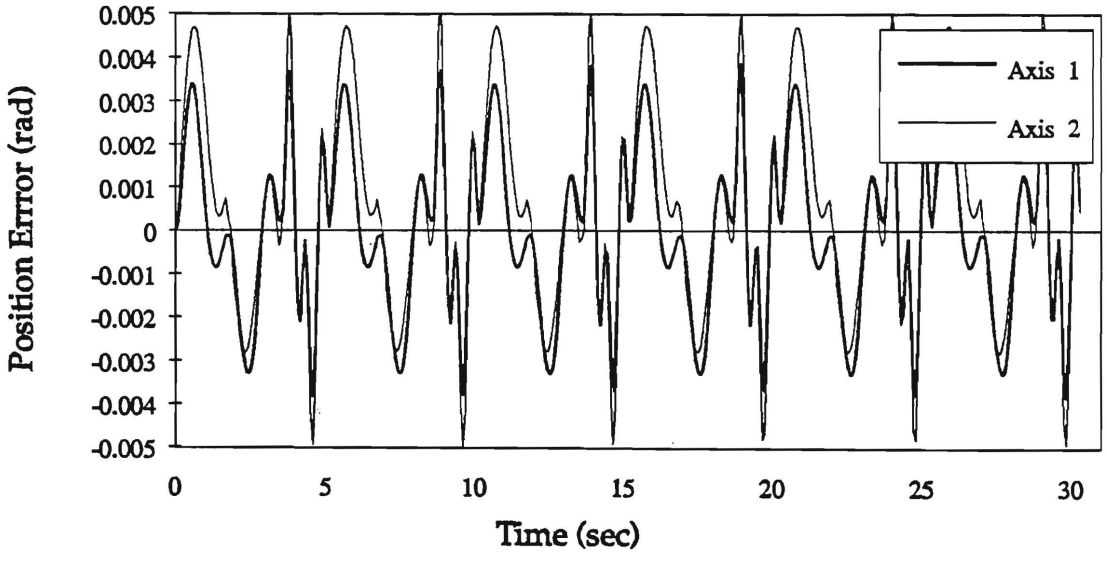

Figure 4.2: PD Feedback Only Joint Position Error

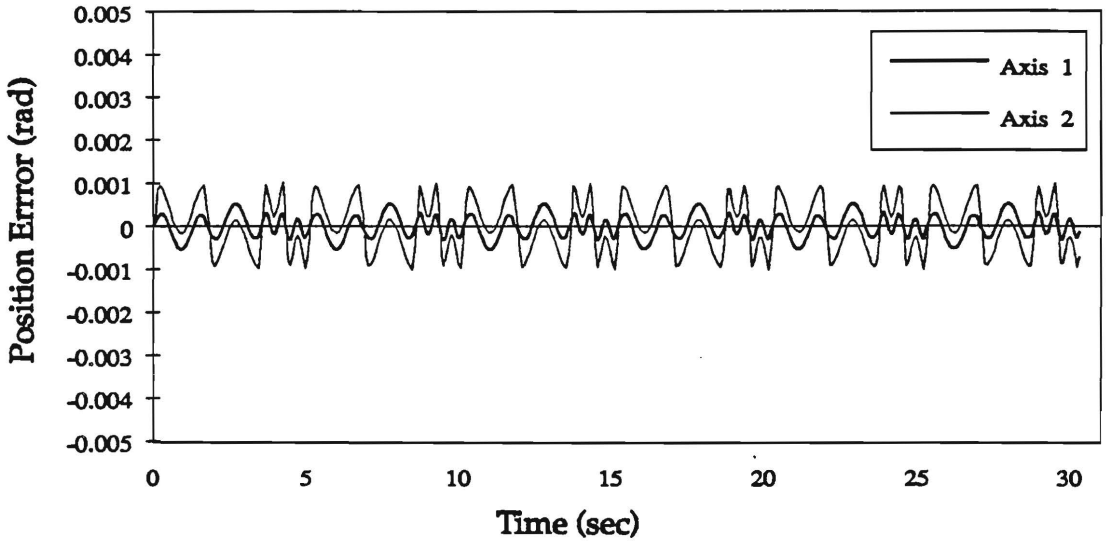

Figure 4.3: DCCL Joint Position Error

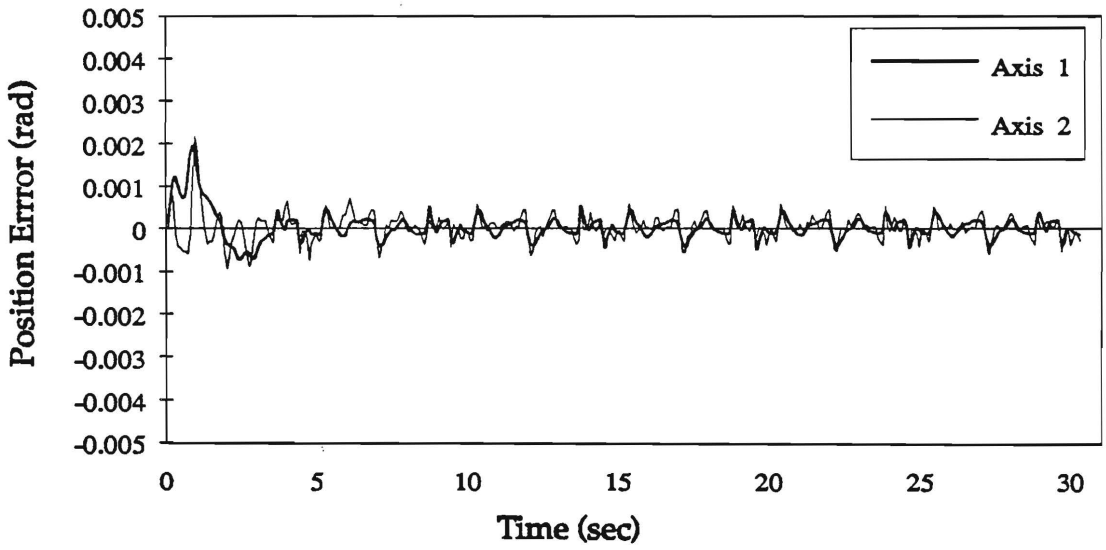

Figure 4.4: DCAL Joint Position Error

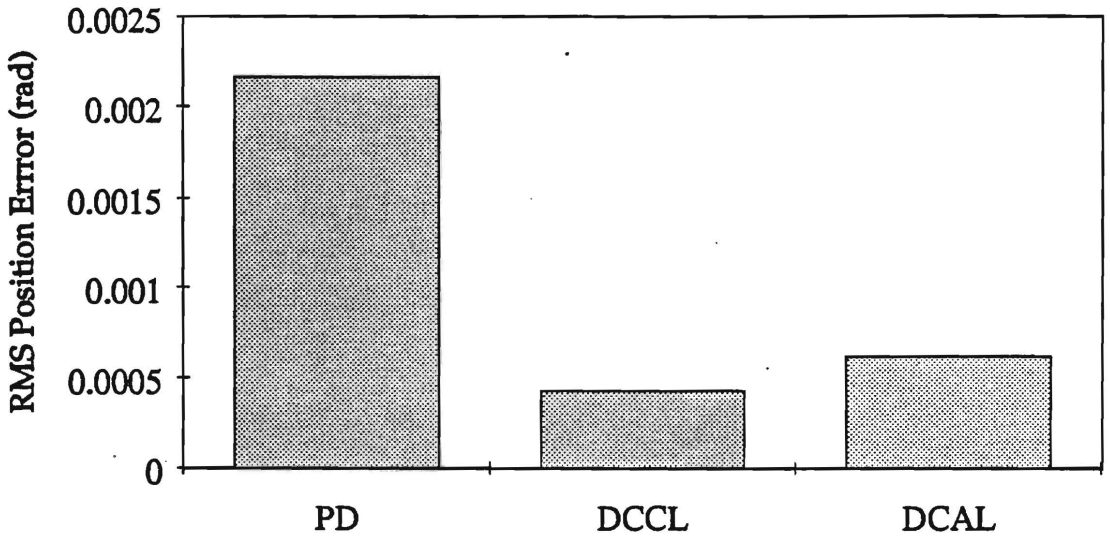

Figure 4.5: First Cycle RMS Joint Position Error

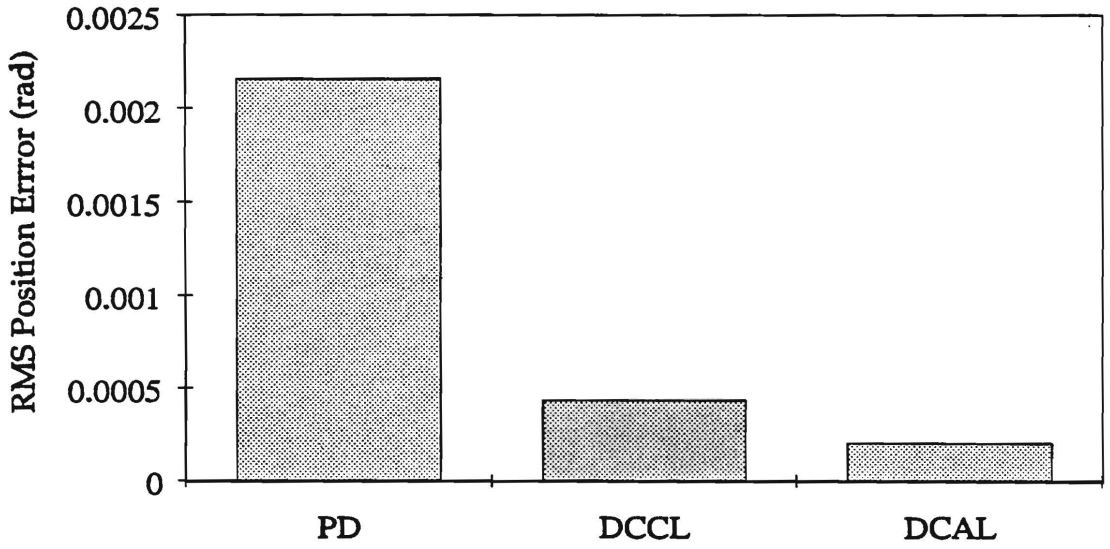

Figure 4.6: Steady State Cycle RMS Joint Position Error

Chapter 5

Desired Compensation Learning Law

5.1 Introduction

In this chapter we will present a *repetitive* learning controller referred to as the *Desired Compensation Learning Law* (DCLL). This algorithm will be formulated in both the *joint* and *Cartesian* space of the manipulator. The Cartesian controller, referred to as the *Desired Compensation Cartesian Learning Law* (DCCLL), will be capable of executing a desired trajectory described in Cartesian coordinates without calculation of any inverse kinematics. This algorithm also forms a foundation for the learning *force* controller to be presented in chapter 8.

Both algorithms use the same feedback/feedforward structure of the DCCL and DCAL presented in the previous chapter. The joint space DCLL does not require any dynamic or kinematic information about the robot to calculate the feedforward signal, and the Cartesian space DCCLL only requires a kinematic description. Both schemes are highly computationally and memory efficient, and are easily implemented on simple, low cost digital hardware.

As previously mentioned, these algorithms are applicable to the control of a robotic

manipulator undergoing a *repetitive* or *periodic* task. By repetitive we mean that the robot is required to repeat some desired task with each cycle requiring the same, finite period of time. Therefore, it is assumed that the desired trajectory signals, $\mathbf{x}_d(t)$, $\dot{\mathbf{x}}_d(t)$, $\ddot{\mathbf{x}}_d(t)$, are periodic with a known period T, i.e.:

$$\begin{aligned} \mathbf{x}_d(t+T) &= \mathbf{x}_d(t) \\ \dot{\mathbf{x}}_d(t+T) &= \dot{\mathbf{x}}_d(t) \\ \ddot{\mathbf{x}}_d(t+T) &= \ddot{\mathbf{x}}_d(t) \end{aligned} \tag{5.1}$$

Note that $\mathbf{x}_d(t)$ may imply a joint space *or* Cartesian space desired trajectory depending on the context of the controller.

5.2 Joint Space DCLL

This is the repetitive learning controller formulated in the joint space of the manipulator. We will be using the same structure of the control law presented in equation 4.27, which we restate here for convenience:

$$\mathbf{q} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_v\mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_d \tag{5.2}$$

where $\hat{\mathbf{w}}_d$ is again an estimate of the desired feedforward torque $\mathbf{w}_d$:

$$\mathbf{w}_d = \mathbf{M}(\mathbf{x}_d)\ddot{\mathbf{x}}_d + \mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \mathbf{V}\dot{\mathbf{x}}_d \tag{5.3}$$

The difference between the control law of equation 4.27 (from the DCAL) and equation 5.2 (from the DCLL) is the way in which the feedforward portion, $\hat{\mathbf{w}}_d$, is calculated. Note that an immediate consequence of equations 5.1 is that the desired feedforward torque, $\mathbf{w}_d$, is also periodic and a continuous function of time. The goal of the DCLL is to "learn" this periodic feedforward term needed in the control law of equation 5.2 as the manipulator cycles. To accomplish this, we will approximate w_{d_j}, the j^{th} component of $\mathbf{w}_d$, by a linear combination of some appropriately selected periodic functions ϕ_i, referred to as *shape functions*. We will

refer to this approximation as $\overline{w}_{d_j}$:

$$\overline{w}_{d_j}(t) = \sum_{i=0}^{N} \Theta_i^j \phi_i(t) \tag{5.4}$$

where Θ_i^j are the unknown coefficients (to be determined by the learning algorithm) of each shape function for each $\overline{w}_{d_j}$, and N is the total number of shape functions which we wish to select.

To make the idea of shape functions precise, we give them a mathematical definition:

Definition 5.2.1 *First, let $C(T)$ denote the space of continuous T-periodic functions. Consider a countable set of linearly independent $\{\phi_i \in C(T)\}$ such that*

1. unity *can be expressed as a linear combination of finitely many ϕ_i's.*
2. *the span of $\{\phi_i\}$ is dense in $C(T)$, that is, for any $\mathbf{w}_d \in C(T)$ and $\varepsilon > 0$, there exist an N and $\Theta_i \in \mathbf{R}^n$ such that:*

$$\sup_{t\in[0,T]} |\mathbf{w}_d(t) - \overline{\mathbf{w}}_d(t)| < \varepsilon \tag{5.5}$$

where $\overline{\mathbf{w}}_d(t)$ is the vector of the $\overline{w}_{d_j}$'s, and Θ_i is the vector of the Θ_i^j's.

Here are some typical examples of shape functions which we can use to approximate the periodic continuous function $\mathbf{w}_d$:

Example 5.2.1

a) Fourier Series Approximation:

$$\phi_i(t) = \begin{cases} \cos\left(2\pi i \frac{t}{T}\right) & \text{if } i \leq m \text{ where } m \equiv \frac{N+1}{2} \\ \sin\left(2\pi(i-m)\frac{t}{T}\right) & \text{if } i > m \end{cases} \tag{5.6}$$

This is a classic choice for a time dependent function of the form of equation 5.4 and offers some advantages in particular situations. However, on-line calculation of the update law becomes computationally expensive as the number of terms becomes large.

b) Piecewise Linear Approximation:

In this example the periodic continuous function is approximated by a piecewise linear function, similar to a finite element approximation. Let the linear function for the j^{th} component of the i^{th} piece be

$$\overline{w}_{d_{j_i}} = c_1(t_l) + c_0(1 - t_l) \tag{5.7}$$

where t_l is the local normalized time (i.e. $0 \leq t_l < 1$) for that particular piece. This particular shape function choice offers the advantage of extremely efficient update law computation since only two coefficients must be updated at any time (i.e. c_0 and c_1). It also has the ability to generate "corners" in the feedforward approximation in order to approximate such nonlinear effects as static friction. Other higher order piecewise continuous functions such as a piecewise quadratic or piecewise cubic approximation are also valid shape functions. These may offer possibly higher performance than the linear version with a small penalty in computational load.

For the purpose of describing the piecewise linear function as a shape function, we will use the following notation:

$$\text{Define } \tau_i \equiv \frac{t}{T}N - i$$

the shape functions $\phi_i(t)$ for $t \in [0, T]$ can be expressed by:

$$\phi_i(t) = \begin{cases} 1 - \tau_i & \text{if } 0 \leq \tau_i < 1 \\ 1 + \tau_i & \text{if } -1 \leq \tau_i < 0 \\ 0 & \text{else} \end{cases} \tag{5.8}$$

and $\phi_i(t + kT) = \phi_i(t), k = 1, 2, 3, \ldots$

c) Piecewise Quadratic Approximation:

In this example the periodic continuous function is approximated by a piecewise quadratic function, similar to the piecewise linear approximation of part (b), but with an added degree of freedom for each piece whose corresponding shape function is a square of time. Let the quadratic function for the $2i^{th}$ piece be

$$\overline{w}_{d_{j_{2i}}} = c_2(t_l^2 - t_l) + c_1(t_l) + c_0(1 - t_l) \tag{5.9}$$

where t_l is the local normal time for that particular piece. In this example only 3 coefficients must be updated at a time (i.e. c_0, c_1, and c_2).

For the purpose of describing this piecewise quadratic function as a shape function, we will use the following notation:

$$\text{Define } \tau_i \equiv \frac{1}{2}\left(\frac{t}{T}N - i\right)$$

the shape functions $\phi_i(t)$ for $t \in [0,T]$ can be expressed by:

$$\phi_i(t) = \begin{cases} 1-\tau_i & \text{if } 0 \le \tau_i < 1 \text{ and } i = 2m \\ \tau_i^2 - \frac{1}{4} & \text{if } -\frac{1}{2} \le \tau_i < \frac{1}{2} \text{ and } i = 2m+1 \\ 1+\tau_i & \text{if } -1 \le \tau_i < 0 \text{ and } i = 2m \\ 0 & \text{else} \end{cases} \tag{5.10}$$

where $m = 1,2,3,\ldots$

and $\phi_i(t+kT) = \phi_i(t), k = 1,2,3,\ldots$

d) Polynomial Approximation:

$$\phi_i(t) = t^i \text{ for } t \in [0,T] \tag{5.11}$$

and $\phi_i(t+kT) = \phi_i(t), k = 1,2,3,\ldots$

Although the polynomial approximation is straightforward, a linear or higher order piecewise continuous shape function such as those described in item (b) or (c) will usually be more computationally efficient and offer higher performance.

We may write the total feedforward term $\hat{\mathbf{w}}_d$ in equation 5.2 as:

$$\hat{\mathbf{w}}_d = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{5.12}$$

where the vector $\hat{\Theta}_i \in \mathbf{R}^n$ is the *estimate* of the coefficient vector Θ_i. These coefficients are updated on-line using the following parameter estimation law:

$$\frac{d}{dt}\hat{\Theta}_i(t) = -\mathbf{K}_{li}\phi_i(t)\mathbf{e}_v(t) \quad \mathbf{K}_{li} > 0 \tag{5.13}$$

Remark: The computations involved in performing the integration of equation 5.13 may be performed in parallel with the remainder of the control algorithm calculations if parallel processing capabilities exist. This means that the sampling time of a digital implementation of the DCLL may be kept on the order of a simple PID control algorithm. Also, choices (b) and (c) of example 5.2.1 have an advantage over choices (a) and (d) in that only two or three parameters (respectively) must be updated at a time. This feature makes the piecewise linear or quadratic shape functions very attractive in a situation where parallel processing is not feasible and sampling times must be kept very small.

Remark: Since the shape functions are linearly independent, the resulting adaptation regression vector, $\phi^T = [\phi_0 \cdots \phi_i \cdots \phi_N]$, is *persistently exciting* over one period of the task repetition, i.e. there exist an $\alpha > 0$ such that:

$$\int_t^{t+T} \phi(\tau)\phi^T(\tau)d\tau > \alpha \mathbf{I} \text{ for any } t > 0 \tag{5.14}$$

This remarkable fact is true even if the desired trajectory is *not* persistently exciting.

In order to utilize previous stability results, we can reparameterize equation 5.12 as

$$\mathbf{W}_l(t)\hat{\Theta} = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{5.15}$$

Applying the control law of equations 5.2 and 5.12, and the parameter estimation law given by equation 5.13 to the manipulator system of equation 3.1, we arrive at the following error dynamic system:

$$\begin{aligned} \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n - \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) + \mathbf{W}_l\tilde{\Theta} + \overline{\mathbf{d}} \end{aligned} \tag{5.16}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda\mathbf{e} \tag{5.17}$$

$$\frac{d}{dt}\tilde{\Theta} = -\mathbf{K}_l\mathbf{W}_l^T\mathbf{e}_v \tag{5.18}$$

where $\tilde{\Theta}$ represents the *parameter estimation error* given by

$$\tilde{\Theta} = \hat{\Theta} - \Theta \tag{5.19}$$

and $\Delta \mathbf{w}(\mathbf{e}_v, \mathbf{e})$ is the same as that given in 4.15. $\overline{\mathbf{d}}$ is a bounded disturbance due to the series approximation truncation error and any other disturbances.

We now state and prove our main stability theorem for the joint space DCLL (see also [33]). First, we prove that when $\overline{\mathbf{d}} = \mathbf{0}$, the system is globally asymptotically and locally exponentially stable.

Theorem 5.2.1 *Due to the natural persistent excitation of the "weighting" matrix $\mathbf{W}_l(t)$ given in equation 5.14, the results of theorems 4.3.1–4.3.3 hold for the error system given by equations 5.16–5.18.*

Proof: The proof of the theorem follows directly from the proofs of theorems 4.3.1–4.3.3. □

Remark Due to the analogous result of theorem 4.3.3 for the DCLL, if the disturbance $\overline{\mathbf{d}}$ is "small" in the sense of the theorem, all the trajectory error signals will also remain "small" provided that the system initially starts inside of a prescribed region. Thus, if the series truncation is the only source of disturbance, the tracking error can be made arbitrarily small by selecting a *larger* number of shape functions. Conversely, the system can be made robust to high frequency disturbances by effectively decreasing the bandwidth of the feedforward signal by selecting *fewer* shape functions. Therefore, a trade off exist between the ability of the feedforward signal to approximate system dynamics and at the same time reject unwanted high frequency disturbances. However, the appropriate number of shape functions is usually easy to determine since the frequency content of the system dynamics and the noise are generally separated by a wide margin. The subsequent implementation results confirm the robustness of the proposed learning algorithm to disturbances such as digital quantization, sensor/actuator noise, and high order dynamics.

5.3 Cartesian Space DCCLL

This is the Cartesian space version of the same repetitive learning controller presented in the previous section. A desired trajectory for this formulation may be expressed, stored,

and executed directly in terms of some Cartesian coordinate system. This is advantageous because a robot trajectory is usually much more easily described in terms of Cartesian coordinates. Also, since *inverse* kinematics are not required, the algorithm is computationally efficient and easy to implement.

To formulate the Cartesian feedback controller, we first need to define the trajectory following position and velocity errors analogous to their joint space counterparts. Define the actual Cartesian position and velocity vectors as $\mathbf{x}_c(t)$ and $\dot{\mathbf{x}}_c(t)$. We will define the Cartesian error and feedback quantities similar to equations 4.5–4.7 by:

$$\mathbf{e} = \mathbf{x}_c - \mathbf{x}_d \tag{5.20}$$

$$\mathbf{v}_c = \dot{\mathbf{x}}_d - \lambda_c \mathbf{e} \qquad \lambda_c > 0 \tag{5.21}$$

$$\mathbf{e}_v = \dot{\mathbf{x}}_c - \mathbf{v}_c \tag{5.22}$$

The actual Cartesian position vector, $\mathbf{x}_c(t)$ is found by using the *forward* kinematics for the manipulator or by using direct Cartesian feedback (from a camera or laser system). The actual Cartesian velocity vector, $\dot{\mathbf{x}}_c(t)$, is found by using the joint positions $\mathbf{x}_j(t)$ to form the manipulator Jacobian, together with the joint velocities $\dot{\mathbf{x}}_j(t)$. The equation relating these quantities is:

$$\dot{\mathbf{x}}_c = \mathbf{J}(\mathbf{x}_j)\dot{\mathbf{x}}_j \tag{5.23}$$

Under the assumption of equation 3.13, we may write the following control and update laws for the DCCLL analogous to equations 5.2, 5.12, and 5.13:

$$\mathbf{q} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_v\mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_d \tag{5.24}$$

$$\hat{\mathbf{w}}_d = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{5.25}$$

$$\frac{d}{dt}\hat{\Theta}_i(t) = -\mathbf{K}_{li}\phi_i(t)\mathbf{e}_v(t) \qquad \mathbf{K}_{li} > 0 \tag{5.26}$$

where $\mathbf{q}$ represents an imaginary Cartesian force/torque vector applied to the end–effector. To determine the actual input to the manipulator actuators, $\mathbf{q}_j$, we use the properties of

the Jacobian matrix: [3,10]:

$$\mathbf{q}_j = \mathbf{J}^T(\mathbf{x}_j)\mathbf{q} \tag{5.27}$$

For compatibility with the previous stability analysis, we once again reparameterize equation 5.25 as

$$\mathbf{W}_l(t)\hat{\Theta} = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{5.28}$$

Applying the control law of equations 5.24 and 5.25, and the parameter estimation law given by equation 5.26 to the manipulator system of equation 3.14, we arrive at the following error dynamic system:

$$\begin{aligned} \mathbf{M}_c(\mathbf{x}_c)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n - \mathbf{C}(\mathbf{x}_c, \dot{\mathbf{x}}_c)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e}) + \mathbf{W}_l\tilde{\Theta} + \overline{\mathbf{d}} \end{aligned} \tag{5.29}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda_c\mathbf{e} \tag{5.30}$$

$$\frac{d}{dt}\tilde{\Theta} = -\mathbf{K}_l\mathbf{W}_l^T\mathbf{e}_v \tag{5.31}$$

where $\tilde{\Theta}$, $\Delta\mathbf{w}(\mathbf{e}_v, \mathbf{e})$, and $\overline{\mathbf{d}}$ are analogous to the quantities defined for the joint space error system (see equations 5.16–5.18).

We now state and prove the stability theorems for the DCCLL.

Theorem 5.3.1 *Under the assumption of equation 3.13, there exists a set of control gains* $\mathbf{K}_p$, $\mathbf{K}_v$, *and* σ_n *such that the ideal (i.e.* $\overline{\mathbf{d}} = \mathbf{0}$*) system described by equations 5.29–5.30 is* locally asymptotically stable.

Proof: The proof of this theorem is completely analogous to the proof of theorem 4.3.1 except that the results are *local* instead of *global* since the manipulator must operate in a subset of the total workspace. This restriction is imposed by equation 3.13 which requires that the robot arm not venture close to a singular configuration. Theorem 4.3.1 states that the tracking errors will converge to zero asymptotically starting from *any* initial condition. The proof of this theorem is identical if the initial conditions are such that equation 3.13 is satisfied, yielding *local* asymptotic stability. □

Theorem 5.3.2 *Due to the natural persistent excitation of the "weighting" matrix* $\mathbf{W}_l(t)$ *(see equation 5.14), and under the assumption of equation 3.13, the results of theorems 4.3.2–4.3.3 hold for the error system given by equations 5.29–5.31.*

Proof: The proof of the theorem follows directly from the proofs of theorems 4.3.2–4.3.3. □

5.4 DCLL Implementation Results

In this section we examine the performance of both the joint space and Cartesian space DCLL repetitive controllers.

5.4.1 DCLL Joint Space Implementation Results

The DCLL joint space scheme was implemented on the IBM 7545 robot for performance comparison to the DCAL. The DCLL easily outperformed the adaptive controller during "steady state" tracking. In fact, the DCLL was able to drive the position error to within the accuracy of the optical encoders after only five cycles of the desired trajectory, and excellent tracking error was attained after only *three* cycles.

The desired trajectory for both controllers was the same as that shown in figure 4.1. The piecewise linear shape function mentioned in example 5.2.1 was used in conjunction with the DCLL.

Figures 5.1 and 5.2 show position error over six cycles of the desired trajectory for the first two (revolute) axes of the IBM 7545. Note that the position error plots are on the same ordinate scale for easy comparison. It can be easily seen that the learning controller (the DCLL) has much superior steady state performance to the DCAL throughout the path period. The DCLL outperforms the DCAL due to its ability to absorb unmodelled dynamic effects in the joints, actuators, and amplifiers, and to cancel unmodelled effects introduced by digital implementation and sampling.

Figure 5.3 shows the RMS (root mean square) position error over the *last cycle* of both controllers. This figure represents an average of the RMS error for the first two axes. Numerical values of the RMS position error are 0.21 *mrad* for the DCAL and 0.036 *mrad* for the DCLL, or an improvement of approximately 82%. The converged error for the DCLL corresponds to less than one encoder line resolution.

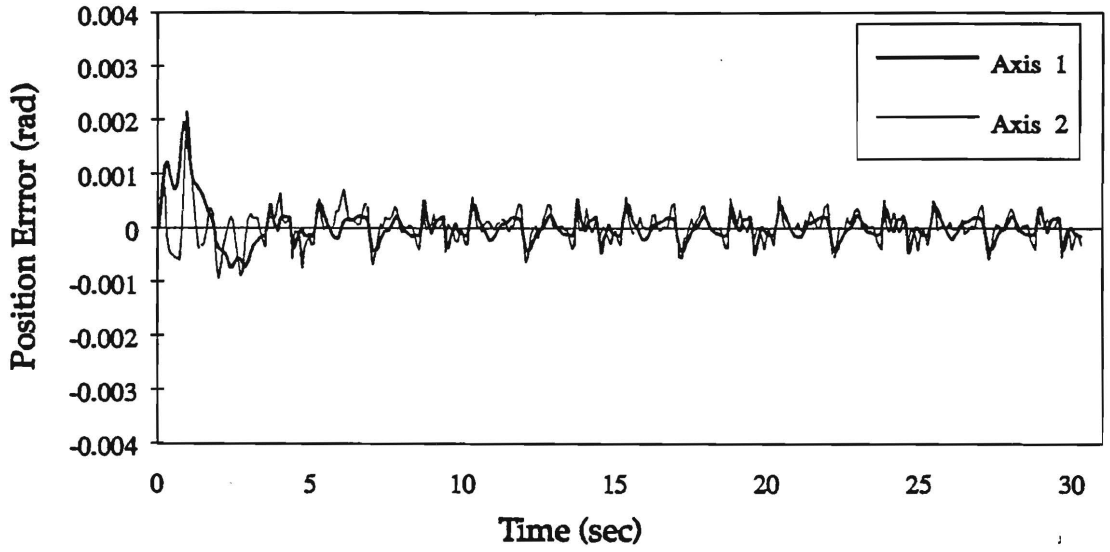

Figure 5.1: DCAL Joint Position Error

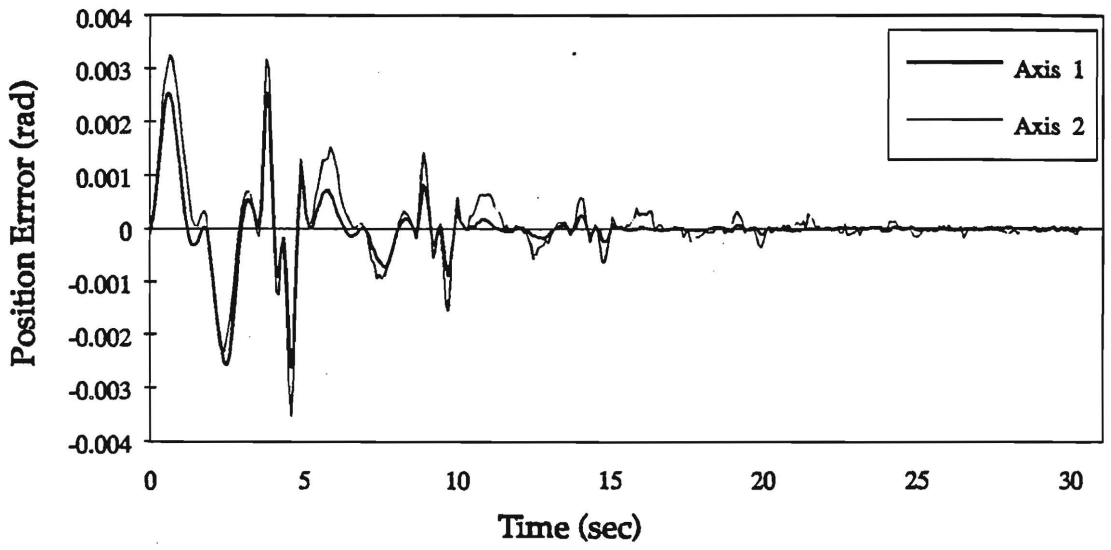

Figure 5.2: DCLL Joint Position Error

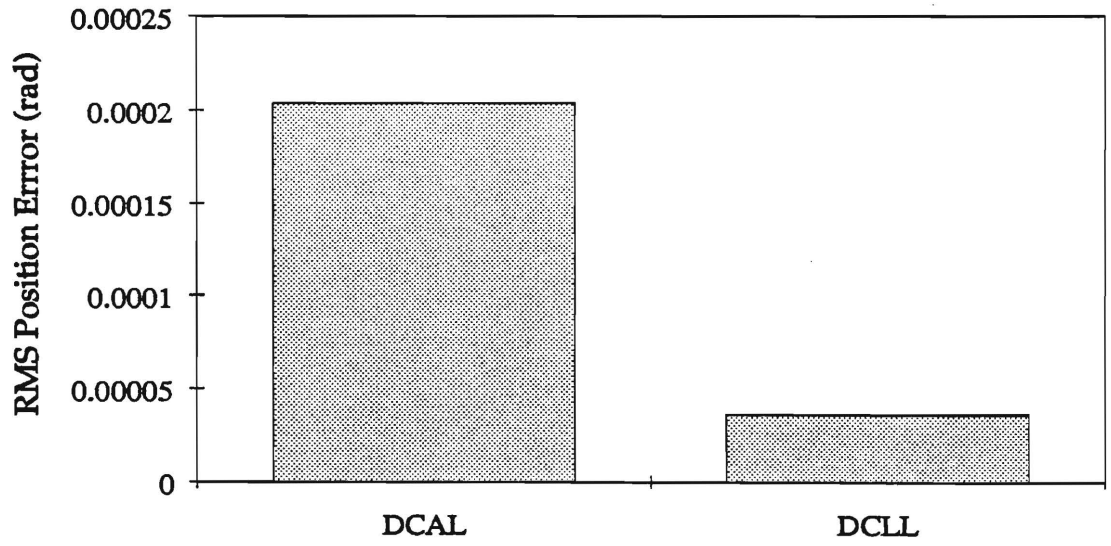

Figure 5.3: Steady State Cycle RMS Joint Position Error

5.4.2 DCCLL Cartesian Space Implementation Results

The DCCLL (the Cartesian space version of the DCLL) was also implemented on the IBM 7545 manipulator and compared to the DCAL scheme. The desired trajectory was prepared using a 7^{th} order polynomial for the x and y position variables. The Cartesian path shape displayed on the horizontal plane of the robot workspace is shown in figure 5.4. The peak velocity reached in the desired trajectory corresponds to the maximum joint

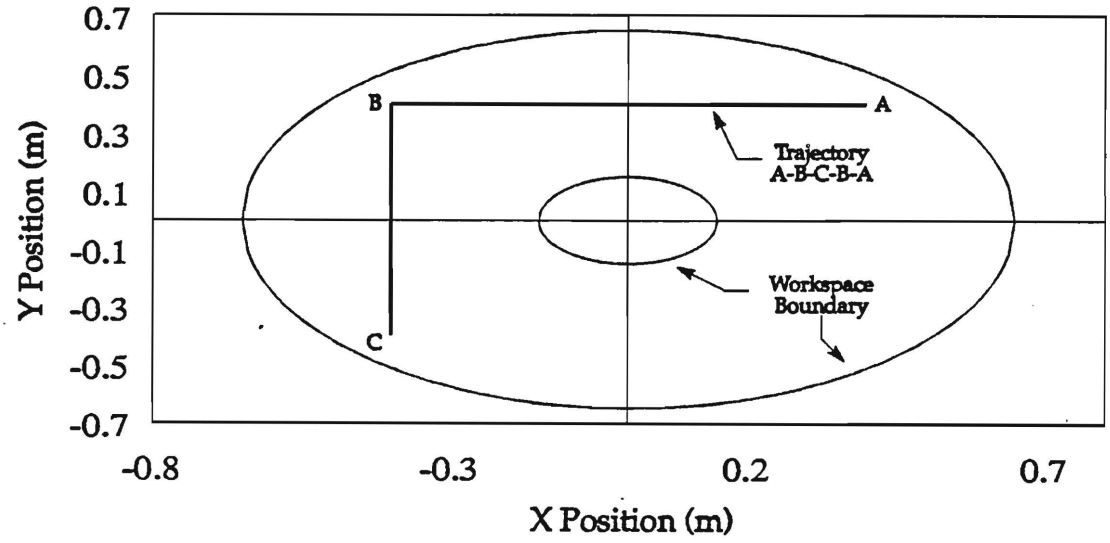

Figure 5.4: Cartesian Space Test Trajectory

velocity specified by the manufacturer for the actuators. The total trajectory cycle time was 7 seconds. The DCCLL was capable of executing the Cartesian space path directly – without the need to perform *inverse* kinematics. The DCAL, however, was implemented for Cartesian space control by using the joint space formulation in conjunction with inverse kinematics. The piecewise linear shape function mentioned in example 5.2.1 was again used in conjunction with the DCCLL.

Figures 5.5 and 5.6 show absolute Cartesian position error in the $x - y$ plane over six cycles of the desired trajectory. In keeping with the concept of presenting the first two axes in the error plots, only the $x - y$ (horizontal) plane errors are shown. The DCCLL shows extremely fast convergence (approximately 3 cycles) and excellent steady state error

as compared to the DCAL. Note that the position error plots are on the same ordinate scale for easy comparison. Once again, the DCCLL outperformed the DCAL by a large margin.

Figure 5.7 shows the RMS (root mean square) of the absolute Cartesian position error over the *last cycle* of both controllers. Numerical values of the RMS position error are $0.104mm$ for the DCAL and $0.0165mm$ for the DCCLL, or an improvement of approximately 84%. Once again, the converged error for the DCCLL corresponds to the equivalent of less than one encoder line resolution.

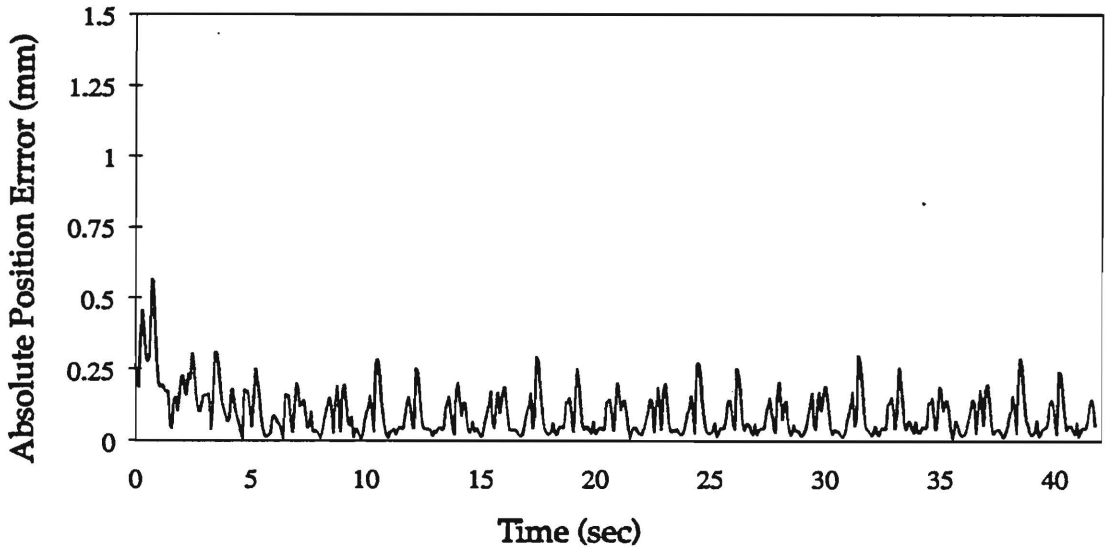

Figure 5.5: DCAL Cartesian Position Error

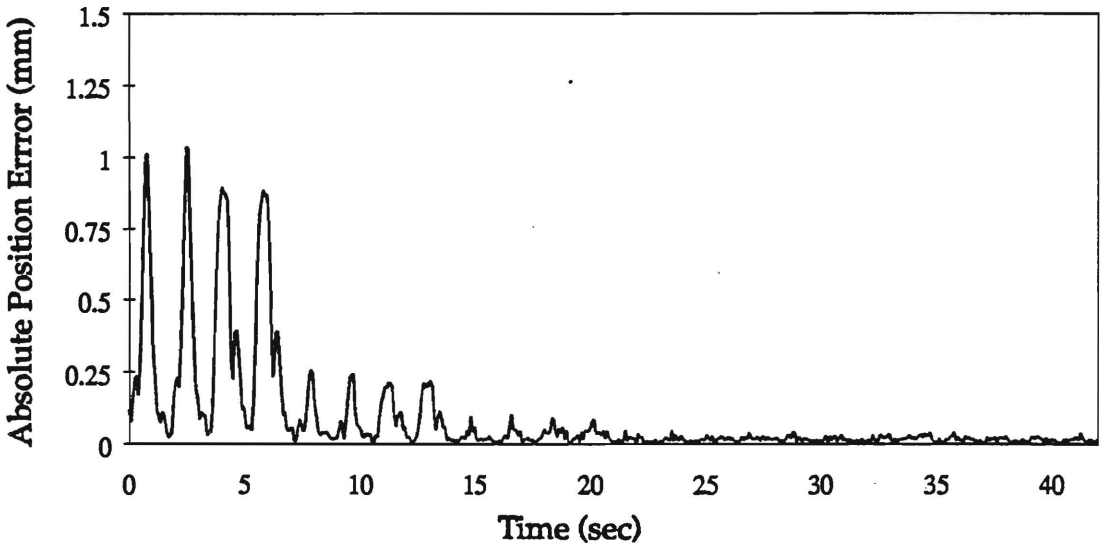

Figure 5.6: DCCLL Cartesian Position Error

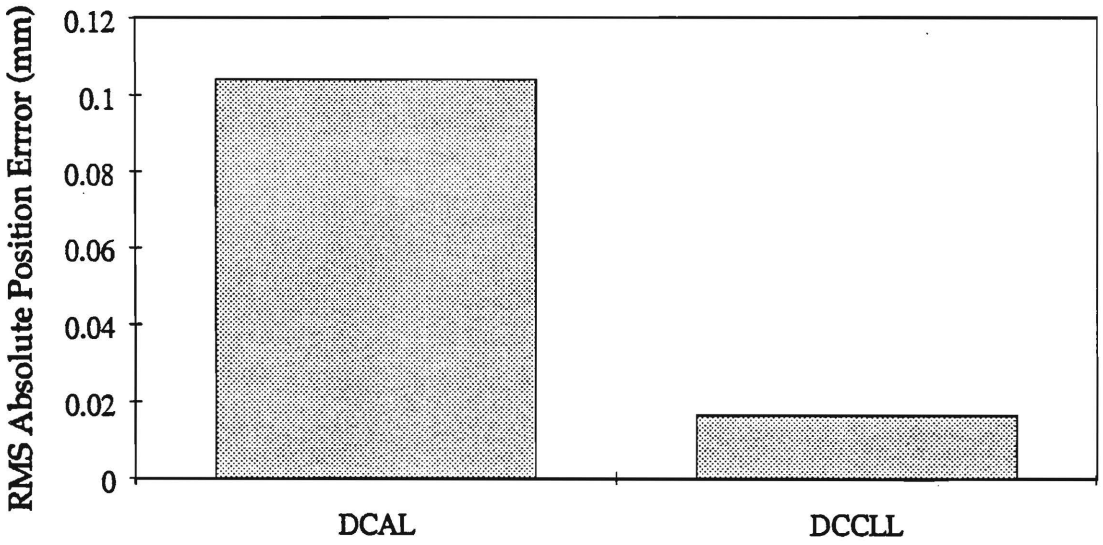

Figure 5.7: Steady State Cycle RMS Cartesian Position Error

Chapter 6

Friction Compensation Adaptive Law

6.1 Introduction

At this point we begin our development of *non–repetitive* learning control laws. The *Friction Compensation Adaptive Law* (FCAL) presented in this chapter is applicable to execution of the same type of non–periodic joint space trajectories as the DCAL. In fact, the FCAL is based on the DCAL with a modification of the way in which friction is accounted for in the system model. To account for friction using the DCAL, the structure of the friction mechanism would have to be explicitly defined, and then unknown coefficients associated with this model would be updated on–line in an attempt to drive the *reference velocity error*, e_v, to zero. The problem, of course, is obtaining an accurate model of the friction. The FCAL circumvents the need to derive an explicit friction model for a system by using a general function capable of describing many different types of friction (linear and nonlinear).

The same concept of estimating an unknown function of *time* from the periodic controllers may be extended to estimating an unknown function of some other independent variable. Specifically in the case of estimating viscous and dry friction, a general function of the joint *velocities* may be appropriate. Certainly the force generated due to viscous

friction is velocity dependent, and dry *dynamic* friction is at the very least dependent on the sign of the velocity. Although a good model of dry *static* friction is also a function of actuator input, it turns out that this dependence is not as important as it might seem.

Before further investigation of the mechanism that the FCAL uses to estimate the friction present in a system, let us examine a typical linear and nonlinear friction model. First rewrite the joint space equation of motion from equation 3.1:

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_j &= \dot{\mathbf{x}}_j \\ \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\dot{\mathbf{x}}_j + \mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{F}(\dot{\mathbf{x}}_j,\mathbf{q}) &= \mathbf{q}+\mathbf{d} \end{aligned} \tag{6.1}$$

where $\mathbf{F}(\dot{\mathbf{x}}_j,\mathbf{q})$ replaces the purely viscous $\mathbf{V}\dot{\mathbf{x}}_j$ term and represents a generalized force due to both viscous and dry (static and dynamic) friction. A reasonable representation for $\mathbf{F}(\dot{\mathbf{x}}_j,\mathbf{q})$ would be:

$$\mathbf{F}(\dot{\mathbf{x}}_j,\mathbf{q}) = \begin{cases} \mathbf{C}\dot{\mathbf{x}}_j + \text{sign}(\dot{\mathbf{x}}_j)\mathbf{F}_d & \text{if } |\dot{\mathbf{x}}_j| > 0 \\ \text{sign}(\mathbf{q})\mathbf{F}_s & \text{if } |\mathbf{q}| > \mathbf{F}_s \text{ and } \dot{\mathbf{x}}_j = 0 \\ \mathbf{q} & \text{if } |\mathbf{q}| \le \mathbf{F}_s \text{ and } \dot{\mathbf{x}}_j = 0 \end{cases} \tag{6.2}$$

where each element of the vector $\mathbf{F}(\dot{\mathbf{x}}_j,\mathbf{q})$ is computed based on the corresponding element of $\dot{\mathbf{x}}_j$ and $\mathbf{q}$. $\mathbf{F}_d$ is a constant vector of Coulomb (dry) dynamic friction forces, $\mathbf{F}_s$ is a constant vector of static friction ("stiction") forces, and $\mathbf{C}$ is the viscous friction coefficient matrix similar to $\mathbf{V}$ in equation 3.1.

Even though 6.2 is a highly nonlinear function of the manipulator joint velocity $\dot{\mathbf{x}}_j$ and actuator input $\mathbf{q}$, the real friction model for a physical system may be much *more* complicated. In fact, the IBM 7545 robot used in this paper for implementation results demonstrated slowly time varying $\mathbf{C}$, $\mathbf{F}_d$, and $\mathbf{F}_s$, and velocity dependent dynamic dry friction coefficients $\mathbf{F}_d$ and viscous friction coefficients $\mathbf{C}$. These additional functional dependencies show that an accurate friction model for many physical systems may be quite difficult to derive and identify. Moreover, these friction effects are usually *not* negligible. Once again using the example of the IBM 7545 robot with high reduction harmonic drives, a large portion of the input for this manipulator is expended overcoming the friction effects mentioned above.

6.2 FCAL Development

The particular version of the FCAL presented here will use a *piecewise linear* function of the manipulator joint velocity, $\dot{\mathbf{x}}_j$, to estimate the total friction force for a particular joint. This will be the same function as described in example 5.2.1, part (b), but with velocity as the independent variable as opposed to time. The other shape functions given in the example are possible – such as a piecewise quadratic or a simple polynomial function – but the piecewise linear function yields excellent results and is simple to implement computationally. Although the friction model presented in equation 6.2 uses both the joint velocity *and* the actuator input as independent variables, we will see that the FCAL's use of a function of only the velocity will yield excellent performance.

We can modify the expression for the manipulator dynamics given in equation 6.1 to explicitly account for friction which is only a function of $\dot{\mathbf{x}}_j$:

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_j &= \dot{\mathbf{x}}_j \\ \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\dot{\mathbf{x}}_j + \mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{F}(\dot{\mathbf{x}}_j) &= \mathbf{q} + \mathbf{d} \end{aligned} \tag{6.3}$$

Now the term $\mathbf{d}$ accounts for any friction not modeled by $\mathbf{F}(\dot{\mathbf{x}}_j)$ and any other disturbances. An approximation of the friction function $\mathbf{F}(\dot{\mathbf{x}}_j)$ is given by:

$$\overline{\mathbf{F}}(\dot{\mathbf{x}}_j) = \sum_{i=0}^{i=N} \Psi_i \phi_i(\dot{\mathbf{x}}_j) \tag{6.4}$$

where Ψ_i is a vector of constant coefficients for each *shape function* $\phi_i(\dot{\mathbf{x}}_j)$.

To show that $\overline{\mathbf{F}}(\dot{\mathbf{x}}_j)$ in 6.4 can approximate $\mathbf{F}(\dot{\mathbf{x}}_j)$ within a prescribed tolerance we give the shape function for the FCAL a mathematical definition similar to that for a periodic system in definition 5.2.1:

Definition 6.2.1 *First, let $C(\dot{x})$ denote a space of piecewise continuous functions whose domain is confined such that $\dot{x} \in [-\dot{x}_{max}, \dot{x}_{max}]$. Consider a countable set of linearly independent $\{\phi_i \in C(\dot{x})\}$ such that*

1. unity *can be expressed as a linear combination of finitely many ϕ_i's.*

2. *the span of* $\{\phi_i\}$ *is dense in* $C(\dot{x})$*, that is, for any* $F(\dot{x}) \in C(\dot{x})$ *and* $\varepsilon > 0$*, there exist an* N *and* $\Psi_i \in \mathbf{R}^n$ *such that:*

$$\sup_{\dot{x} \in [-\dot{x}_{max}, \dot{x}_{max}]} \left| F(\dot{x}) - \overline{F}(\dot{x}) \right| < \varepsilon \tag{6.5}$$

where $\overline{F}(\dot{x})$ *is the approximation of the friction function for a single axis given by equation 6.4.*

This simply means that we may approximate the velocity dependent friction function $\mathbf{F}(\dot{\mathbf{x}}_j)$ as closely as we wish using equation 6.4.

The FCAL will use the same form of control law as the DCAL:

$$\mathbf{q} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_v\mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_{df} \tag{6.6}$$

All of the terms are the same as those in equation 4.27 with the exception of the feedforward estimate, $\hat{\mathbf{w}}_{df}$. For the FCAL, the purely viscous friction term $\mathbf{V}\dot{\mathbf{x}}_d$ in $\mathbf{w}_d$ from the DCAL has been replaced with the more general friction estimation function.

The *desired* value of $\hat{\mathbf{w}}_{df}$ is given by

$$\mathbf{w}_{df} = \mathbf{M}_j(\mathbf{x}_d)\ddot{\mathbf{x}}_d + \mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d) \tag{6.7}$$

As in the formulation of the DCAL, we can reparameterize the previous equation by separating the functional portion from the constant parameters in the following way:

$$\mathbf{w}_{df} = \mathbf{W}_f(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{p} \tag{6.8}$$

where $\mathbf{p}$ is the vector containing both the dynamic parameters associated with $\mathbf{M}_j(\mathbf{x}_d)$, $\mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)$, and $\mathbf{g}(\mathbf{x}_d)$, and the friction function coefficients Ψ_i.

As mentioned previously, only a piecewise linear approximation of the friction function will be presented in this paper. Therefore, the explicit form of the shape functions, ϕ_i, for this type of friction estimation are given in the example below. Note that the other types of *time* dependent shape functions given in example 5.2.1 have direct analogies to the *velocity* dependent forms.

Example 6.2.1 Piecewise Linear Friction Approximation:

In this example the velocity dependent function $\overline{\mathbf{F}}(\dot{\mathbf{x}}_d)$ is approximated by a piecewise linear function, similar to a finite element approximation. Let the linear function for the i^{th} piece be

$$\overline{F}_j = c_i(\dot{x}_{jl}) + c_{i-1}(1 - \dot{x}_{jl}) \tag{6.9}$$

where $\dot{x}_{jl}$ is the local normalized velocity for the l^{th} piece for a particular axis j. For the purpose of describing this piecewise linear function as a shape function, we will use the following notation:

$$\text{Define } \tau_i \equiv \frac{\dot{x}_j}{\dot{x}_{jmax}} N - i$$

The shape function for a particular axis $\phi_{ij}(\dot{x}_j)$ for $\dot{x}_j \in [-\dot{x}_{jmax}, \dot{x}_{jmax}]$ can be expressed by:

$$\phi_{ij}(\dot{x}_j) = \begin{cases} 1 - \tau_i & \text{if } 0 \leq \tau_i < 1 \\ 1 + \tau_i & \text{if } -1 \leq \tau_i < 0 \\ 0 & \text{else} \end{cases} \tag{6.10}$$

The i^{th} shape function ϕ_i represented in equation 6.4 is simply the vector of all ϕ_{ij} shape functions for each axis j of the manipulator.

Keep in mind that the particular formulation of the piecewise linear shape function given in the example is necessary for compatibility with the structure of equation 6.4. However, this type of formulation would never be used in an implementation of the controller because of the simplicity of directly updating the piecewise linear coefficients using equation 6.9.

An estimate of the feedforward friction $\overline{\mathbf{F}}(\dot{\mathbf{x}}_d)$ is given by:

$$\hat{\mathbf{F}}(\dot{\mathbf{x}}_d) = \sum_{i=0}^{i=N} \hat{\Psi}_i \phi_i(\dot{\mathbf{x}}_d) \tag{6.11}$$

and an estimate of the feedforward torque $\mathbf{w}_{df}$ is given by:

$$\hat{\mathbf{w}}_{df} = \mathbf{W}_f(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\hat{\mathbf{p}} \tag{6.12}$$

where $\hat{\mathbf{p}}$ is an estimate of the parameter vector $\mathbf{p}$ which includes the friction estimation coefficients $\hat{\Psi}_i$. Once again the following standard parameter update law with constant

adaptation gains is used:

$$\frac{d}{dt}\hat{\mathbf{p}} = -\mathbf{K}_a \mathbf{W}_f{}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_v \qquad \mathbf{K}_a > 0 \tag{6.13}$$

Applying the control law of equation 6.6 and the update law of equation 6.13 to equation 6.3, we obtain the following error dynamics:

$$\begin{aligned} \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) - \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}_f(\mathbf{e}_v, \mathbf{e}) + \mathbf{W}_f(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\tilde{\mathbf{p}} + \overline{\mathbf{d}} \end{aligned} \tag{6.14}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda\mathbf{e} \tag{6.15}$$

$$\frac{d}{dt}\tilde{\mathbf{p}} = -\mathbf{K}_a \mathbf{W}_f{}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_v \tag{6.16}$$

where $\mathbf{q}_n(\mathbf{e}_v, \mathbf{e})$ is the same as that defined for the DCAL, and $\tilde{\mathbf{p}}$ is defined as:

$$\tilde{\mathbf{p}} = \hat{\mathbf{p}} - \mathbf{p} \tag{6.17}$$

$\overline{\mathbf{d}}$ is a lumped, bounded disturbance which includes the disturbances contained in $\mathbf{d}$ from equation 6.3 as well as any additional disturbance due to using the approximation of equation 6.4 for $\mathbf{F}(\dot{\mathbf{x}}_d)$:

$$\overline{\mathbf{d}} = \mathbf{d} + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d) - \mathbf{F}(\dot{\mathbf{x}}_d) \tag{6.18}$$

$\Delta\mathbf{w}_f(\mathbf{e}_v, \mathbf{e})$ is a state dependent disturbance due to using the *desired* trajectory quantities instead of their *true* counterparts, and is defined by:

$$\begin{aligned} \Delta\mathbf{w}_f &= \left[\mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{v}_j + \mathbf{C}(\mathbf{x}_j, \dot{\mathbf{x}}_j)\mathbf{v}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{F}(\dot{\mathbf{x}}_j)\right] \\ &\quad - \left[\mathbf{M}_j(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \mathbf{C}(\mathbf{x}_d, \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \mathbf{F}(\dot{\mathbf{x}}_d)\right] \end{aligned} \tag{6.19}$$

We now present the following stability theorem for the FCAL:

Theorem 6.2.1 *Under the assumption of a persistently exciting $\mathbf{W}_f(t)$ (see equation 4.36), the results of theorems 4.3.1–4.3.3 hold for the error system described by equations 6.14–6.16.*

Proof: The proof of the theorem follows directly from the proofs of theorems 4.3.1–4.3.3. □

6.3 FCAL Implementation Results

The FCAL was implemented on the IBM 7545 manipulator for joint space position control. In comparison to the DCAL with only viscous friction estimation, the FCAL showed *extremely* fast convergence and much better steady state error. In fact, the FCAL steady state error corresponded to less than two optical encoder line resolutions.

The adaptive gains for update of the non–friction associated terms (i.e. the parameters associated with **M**, **C**, and **g**) as well as the PD feedback gains were the same for the DCAL and FCAL. Therefore, the difference between the performance of the DCAL and the FCAL can be completely attributed to the way in which the different friction terms model the actual friction. The joint space desired trajectory for both controllers was the same as that shown in figure 4.1, and the piecewise linear shape function described in example 6.2.1 was used in conjunction with the FCAL's friction estimation portion.

Figures 6.1 and 6.2 show position error over six cycles of the desired trajectory for the first two (revolute) axes. The FCAL demonstrates extremely fast convergence during the first few seconds of the first trajectory cycle. The peak error during this time is less than 60% of that for the DCAL, and at steady state the FCAL shows excellent tracking with peak errors near the resolution capabilities of the position feedback system. Note that these position error plots are on the same ordinate scale for easy comparison.

Figures 6.3 and 6.4 show the RMS (root mean square) position error over the *first* and *last* cycles of the two controllers. These figures represent an average of the RMS error for the first two axes. Note that the *first* cycle RMS error for the FCAL is less than 50% of that for the DCAL, and during the "steady state" or "converged" cycle (i.e. the *last* cycle), the FCAL betters the RMS position error of the DCAL by 66%. The actual numerical values for the last cycle RMS error are 0.21 *mrad* for the DCAL and 0.070 *mrad* for the FCAL. The converged error for the FCAL corresponds to less than two encoder line resolutions.

Figure 6.5 shows the velocity dependent friction function as estimated by the FCAL for the first two axes of the IBM 7545 robot. Note the piecewise linear shape of the curves and

the "interpolation" of stiction effects near $\dot{\mathbf{x}} = \mathbf{0}$. Although this function looks very erratic, is must be considered "correct" because of the excellent performance of the FCAL. This fact is confirmed by input/output friction identification data taken on the robot. Although not directly addressed in this dissertation, the FCAL could be used as a friction identification tool if the desired input was persistently exciting. Also, the FCAL is naturally capable of absorbing linear and nonlinear velocity dependent actuator dynamics since these effects "look" just like friction from an input/output view of the system.

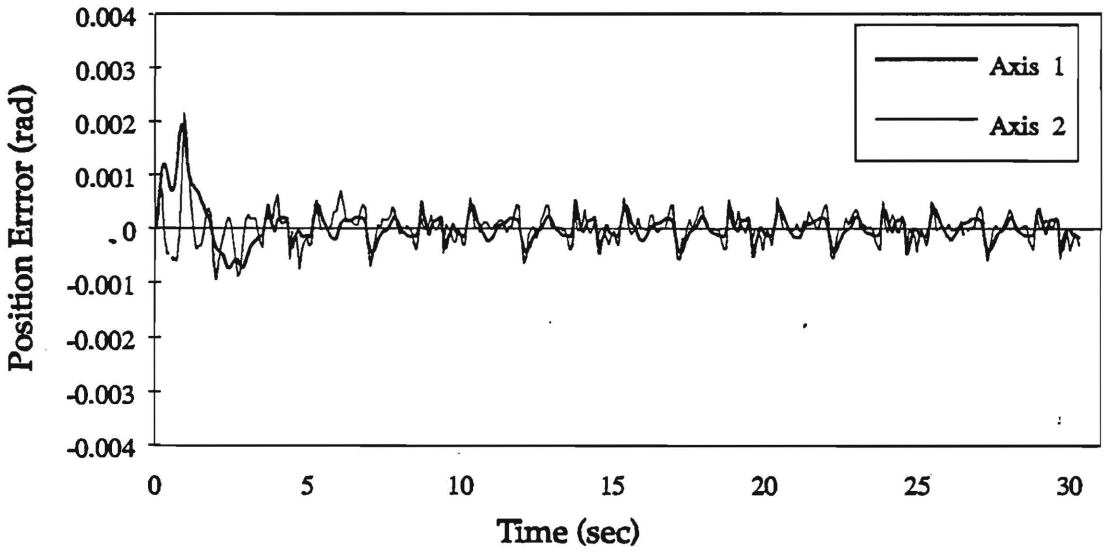

Figure 6.1: DCAL Joint Position Error

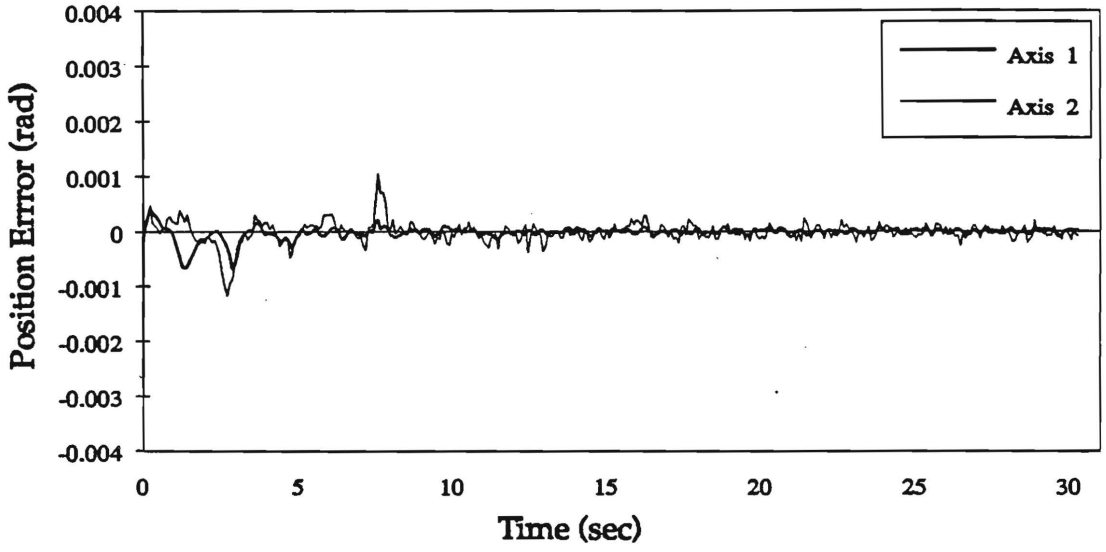

Figure 6.2: FCAL Joint Position Error

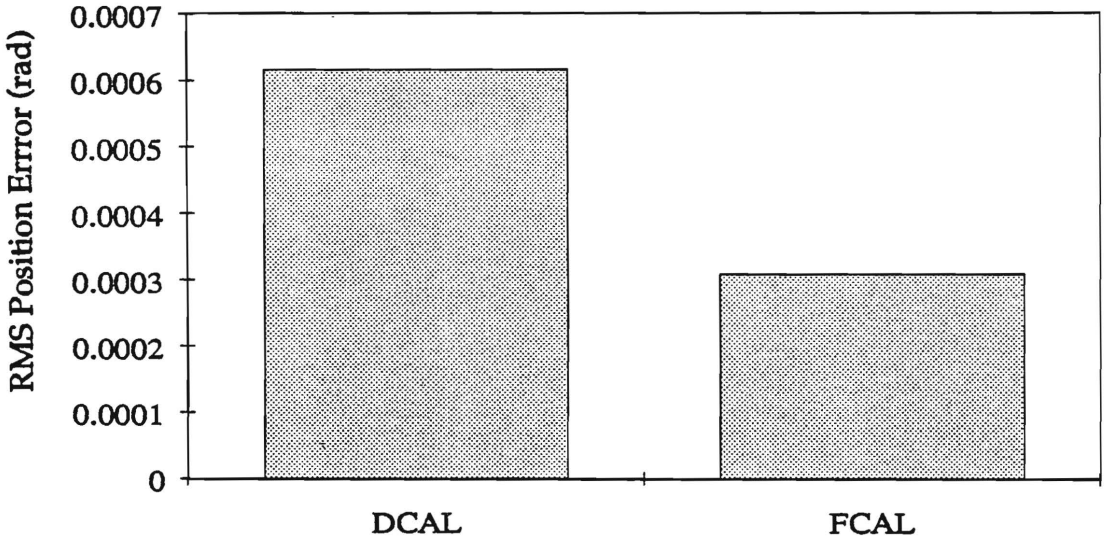

Figure 6.3: First Cycle RMS Joint Position Error

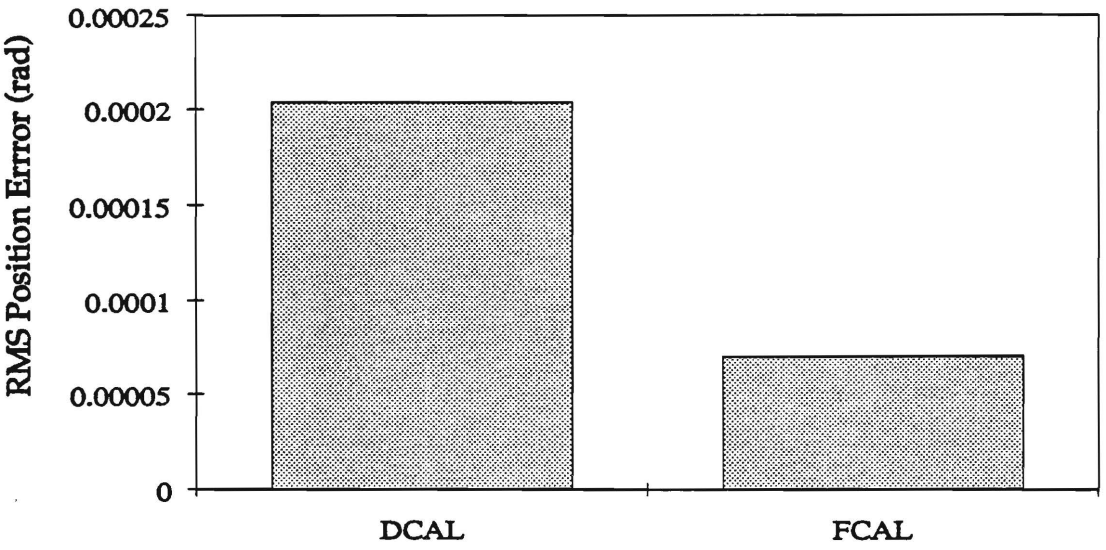

Figure 6.4: Steady State Cycle RMS Joint Position Error

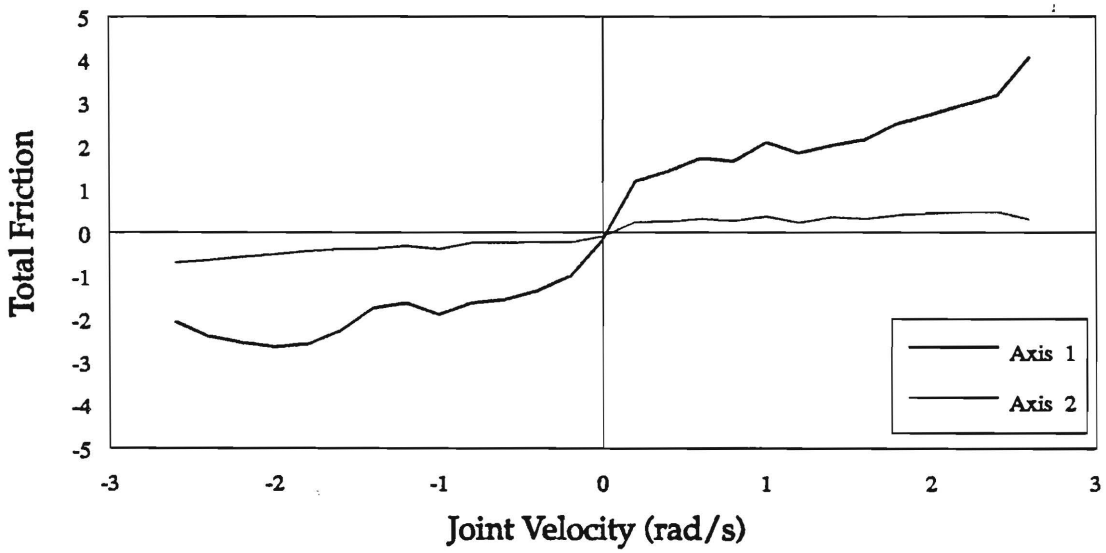

Figure 6.5: FCAL Friction Function Estimate

Chapter 7

Configuration Learning Law

7.1 Introduction

In this chapter we present a learning control algorithm with performance similar to that of the FCAL, but without the requirement of deriving the specific equations of motion for the manipulator. More precisely, we will use a finite element–like approximation of the general dynamic equations for a robot. By using this estimate for the *position* dependent portions of the equation of motion, we will generate hypersurfaces whose independent coordinates are the joint positions of the robot. Therefore, the shape of these hypersurfaces will be a function of the *configuration* of the open kinematic chain. For this reason we will refer to the algorithm as the *Configuration Learning Law* (CLL).

The CLL will be applicable to joint space control of a manipulator performing a repetitive or non–repetitive task. This controller will again use the same feedback/feedforward structure of the FCAL and DCLL presented in previous chapters. Although this scheme resembles a neural network in appearance, it is much more computationally efficient and easily implemented on today's low cost digital hardware. Memory considerations, however, are more critical. Since we will be generating hypersurfaces of several dimensions, memory requirements can quickly multiply. For this reason, we will restrict the approximation to at most the first *three* links of the manipulator. This is usually completely satisfactory since the

first three links are by far the most massive and dynamically coupled on many industrial 6 DOF robots. The last three links usually consist of a triple intersecting axes revolute wrist. This portion of the robot does not experience the large nonlinear and coupled dynamic effects of the first three links.

By assuming that the last 3 links of a 6 DOF manipulator are decoupled, the configuration space is reduced to three 1–dimensional subspaces plus a single 3–dimensional subspace. As an example of the memory requirements for a specific application, consider the IBM 7545 4–DOF robot. The last 2 links are completely dynamically decoupled so that the total configuration space for 25 discrete quantities of each joint is $25^2 + 25 + 25 = 675$ "nodes." This means that only 675 memory elements are required for each approximation function within the equation of motion. This corresponds to a total memory requirement of less than 2Kbytes for the entire configuration feedforward term. Given this example, it can be seen that the CLL is easily implemented with inexpensive memory requirements.

7.2 CLL Development

Before presenting the CLL's control law, let us take a closer look at the joint space equations of motion for a robotic manipulator (see equation 6.3)

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_j &= \dot{\mathbf{x}}_j \\ \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\dot{\mathbf{x}}_j + \mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j + \mathbf{g}(\mathbf{x}_j) + \mathbf{F}(\dot{\mathbf{x}}_j) &= \mathbf{q} + \mathbf{d} \end{aligned} \tag{7.1}$$

where we have included the velocity dependent friction term $\mathbf{F}(\dot{\mathbf{x}}_j)$ from the FCAL. Note that this equation is only a function of $\mathbf{x}_j$, $\dot{\mathbf{x}}_j$, and $\frac{d}{dt}\dot{\mathbf{x}}_j$, which are n dimensional vectors. Theoretically, a feedforward term $\mathbf{w}_d$ based on this equation could be formed for any point in the n^3 dimensional space of the robot. This is precisely what a neural network might attempt to do – but current reasonable memory and computational limits prevent this as previously mentioned. Instead, note that several of the terms have a dependence on the position vector, $\mathbf{x}_j$. Even the Coriolis term $\mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j$ may be rewritten in a general form

by using equation 3.2 so that its structure is completely defined by the mass matrix (which is only a function of position):

$$\overline{\mathbf{C}}(\mathbf{M}_j(\mathbf{x}_j),\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j = \mathbf{C}(\mathbf{x}_j,\dot{\mathbf{x}}_j)\dot{\mathbf{x}}_j \tag{7.2}$$

By separating out the position dependent portions of equation 7.1, we can rewrite the *desired* FCAL feedforward signal in the following form

$$\mathbf{w}_d = \mathcal{F}_1(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \overline{\mathbf{C}}(\mathcal{F}_1(\mathbf{x}_d),\dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathcal{F}_2(\mathbf{x}_d) + \mathbf{F}(\dot{\mathbf{x}}_d) \tag{7.3}$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ are position dependent matrices which replace $\mathbf{M}_j(\mathbf{x}_d)$ and $\mathbf{g}(\mathbf{x}_d)$ respectively. By formulating the feedforward input in this way, an approximation of $\mathbf{w}_d$ may be generated which requires that learning be performed only over the space of n dimensions of $\mathbf{x}_d$ (i.e. the configuration space of the manipulator) and the one dimensional n spaces of $\dot{\mathbf{x}}_d$ (as in the FCAL):

$$\overline{\mathbf{w}}_d = \overline{\mathcal{F}}_1(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \overline{\mathbf{C}}(\overline{\mathcal{F}}_1(\mathbf{x}_d),\dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \overline{\mathcal{F}}_2(\mathbf{x}_d) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d) \tag{7.4}$$

In the spirit of the previous learning controllers, the following approximations are used for a *three* degree of freedom system:

$$\overline{\mathcal{F}}_1(\mathbf{x}_d) = \sum_{i,j,k=0}^{M} \Theta_{1_{ijk}}\varphi_{ijk}(\mathbf{x}_d) \tag{7.5}$$

$$\overline{\mathcal{F}}_2(\mathbf{x}_d) = \sum_{i,j,k=0}^{M} \Theta_{2_{ijk}}\varphi_{ijk}(\mathbf{x}_d) \tag{7.6}$$

$$\overline{\mathbf{F}}(\dot{\mathbf{x}}_d) = \sum_{i=0}^{N} \Psi_i\phi_i(\dot{\mathbf{x}}_d) \tag{7.7}$$

where the summation notation is defined as

$$\sum_{i,j,k=0}^{M} \equiv \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{k=0}^{M} \tag{7.8}$$

$\Theta_{1_{ijk}}$ is a 3 by 3 *symmetric* matrix, and $\Theta_{2_{ijk}}$ is a 3 by 1 vector. The $\varphi_{ijk}(\mathbf{x}_d)$ terms are the multidimensional *shape functions* for the CLL. $\overline{\mathcal{F}}_1(\mathbf{x}_d)$ (and therefore $\Theta_{1_{ijk}}$) is symmetric because it replaces the symmetric mass matrix $\mathbf{M}_j(\mathbf{x}_d)$. This is a general simplification for

an open kinematic chain robot, and does not require specific structure in the equations of motion.

$\overline{\mathbf{F}}$ in equation 7.7 has already been shown to approximate $\mathbf{F}$ within a prescribed tolerance in definition 6.2.1. We may now state an analogous definition for $\mathcal{F}_1$ and $\mathcal{F}_2$:

Definition 7.2.1 *First, let* $C(\mathbf{x})$ *denote a space of piecewise continuous functions whose domain is confined to a compact subset of* $\mathbf{R}^3$*:*

$$\mathbf{x} \in D_x = \{[x_{1_{min}}, x_{1_{max}}] \times [x_{2_{min}}, x_{2_{max}}] \times [x_{3_{min}}, x_{3_{max}}]\}$$

where $\times$ *denotes the Cartesian product. Consider a countable set of linearly independent* $\{\varphi_{ijk} \in C(\mathbf{x})\}$ *such that*

1. unity *can be expressed as a linear combination of finitely many* φ_{ijk}*'s.*
2. *the span of* $\{\varphi_{ijk}\}$ *is dense in* $C(\mathbf{x})$*, that is, for any* $\mathcal{F}(\mathbf{x}) \in C(\mathbf{x})$ *and* $\varepsilon > 0$*, there exist an* M *and* θ_{ijk} *such that:*

$$\sup_{\mathbf{x} \in D_x} \left| \mathcal{F}(\mathbf{x}) - \overline{\mathcal{F}}(\mathbf{x}) \right| < \varepsilon \tag{7.9}$$

where $\overline{\mathcal{F}}(\mathbf{x})$ *is any one of the elements in equations 7.5 - 7.6.*

Again, this simply means that we can estimate any term in equations 7.5 – 7.6 as closely as we wish by using this type of approximation.

The CLL will use the same form of control law as the FCAL:

$$\mathbf{q} = -\mathbf{K}_p\mathbf{e} - \mathbf{K}_v\mathbf{e}_v - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) + \hat{\mathbf{w}}_d \tag{7.10}$$

where $\hat{\mathbf{w}}_d$ is an estimate of the required feedforward signal given by equation 7.4 and defined later in equation 7.22. At this point an example of the type of *shape functions* we will be using is in order.

Example 7.2.1 First Order Triangular Approximation:

In this example the elements of the position dependent functions $\mathcal{F}_1(\mathbf{x}_d)$ *and* $\mathcal{F}_2(\mathbf{x}_d)$ *are approximated by a multidimensional piecewise continuous first order function, similar to a*

finite element approximation. Since these functions will only have 2 independent variables (i.e. only 2 dimensions) when implemented on the IBM 7545's first two axes, we will develop the shape functions here in the same form. This formulation is also much easier to physically visualize since a 3 dimensional function would have a 4^{th} dependent dimension.

First, we will divide the two dimensional configuration space *of the manipulator into equally sized right triangles of area A (see figure 7.1). The height of each "node" $P_{1\rightarrow 3}$ will be the value of parameter θ_{ij} where the "i" index corresponds to the dependent variable x_1, and the "j" index corresponds to the dependent variable x_2. For some point P in the interior of area A, the triangle is subdivided into 3 areas A_1, A_2, and A_3 as shown in the figure. We may normalize the length of the legs of the triangle to unity and define normal*

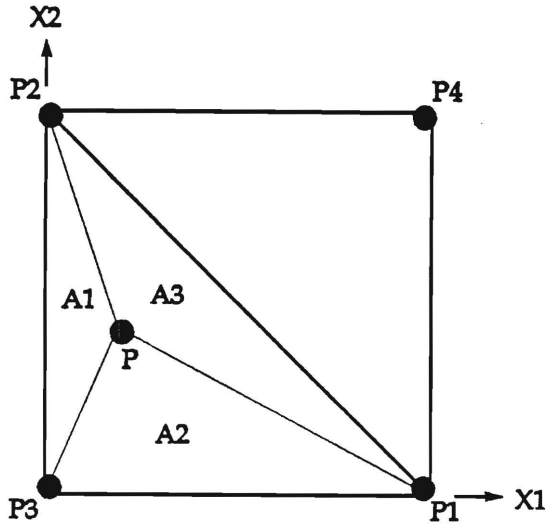

Figure 7.1: CLL Shape Function Diagram

coordinates $\overline{x}_1$ and $\overline{x}_2$ along the legs such that $0 \leq \overline{x}_1 < 1$ and $0 \leq \overline{x}_2 < 1$. The expression for the areas then becomes:

$$A = \frac{1}{2} \tag{7.11}$$

$$A_1 = \frac{1}{2}\overline{x}_1 \tag{7.12}$$

$$A_2 = \frac{1}{2}\overline{x}_2 \tag{7.13}$$

$$A_3 = \frac{1}{2}(1 - \bar{x}_1 - \bar{x}_2) \tag{7.14}$$

Now the height of point P may be expressed in terms of the height of the other nodes and the normal coordinates of P

$$P = P_1\frac{A_1}{A} + P_2\frac{A_2}{A} + P_3\frac{A_3}{A} \tag{7.15}$$

or

$$P = P_1\bar{x}_1 + P_2\bar{x}_2 + P_3(1 - \bar{x}_1 - \bar{x}_2) \tag{7.16}$$

Note that there is a similar triangle B opposite to A. Triangle A is defined for $\bar{x}_2 \leq 1 - \bar{x}_1$, *triangle B is defined for* $\bar{x}_2 > 1 - \bar{x}_1$, *and the height of a point P within B is given by*

$$P = P_1(1 - \bar{x}_2) + P_2(1 - \bar{x}_1) + P_4(\bar{x}_1 + \bar{x}_2 - 1) \tag{7.17}$$

As in the previous shape function examples, we reformulate the development above for compatibility with equations 7.5 - 7.6

$$\begin{aligned} \text{Define } \tau_i &\equiv \frac{x_1}{\Delta x_1}M - i \\ \tau_j &\equiv \frac{x_2}{\Delta x_2}M - j \end{aligned}$$

where $\Delta x_l \equiv x_{lmax} - x_{lmin}$. *The shape function for a particular* $\varphi_{ij}(\mathbf{x})$ *for* $\mathbf{x} \in D_x$ *can be expressed by:*

$$\varphi_{ij}(\mathbf{x}) = \begin{cases} 1 - \tau_i - \tau_j & \text{if } 0 \leq \tau_i < +1 \text{ and } 0 \leq \tau_j < +1 \text{ and } \tau_j \leq 1 - \tau_i \\ 1 + \tau_i & \text{if } -1 \leq \tau_i < 0 \text{ and } 0 \leq \tau_j < +1 \text{ and } \tau_j \leq -\tau_i \\ 1 + \tau_j & \text{if } 0 \leq \tau_i < +1 \text{ and } -1 \leq \tau_j < 0 \text{ and } \tau_j \leq -\tau_i \\ 1 + \tau_i + \tau_j & \text{if } -1 \leq \tau_i < 0 \text{ and } -1 \leq \tau_j < 0 \text{ and } \tau_j > -1 - \tau_i \\ 1 - \tau_j & \text{if } -1 \leq \tau_i < 0 \text{ and } 0 \leq \tau_j < +1 \text{ and } \tau_j > -\tau_i \\ 1 - \tau_i & \text{if } 0 \leq \tau_i < +1 \text{ and } -1 \leq \tau_j < 0 \text{ and } \tau_j > -\tau_i \\ 0 & \text{else} \end{cases} \tag{7.18}$$

Note that equation 7.18 would *not* be used in an implementation situation for calculation of φ_{ij}. Instead, equations 7.16 and 7.17 would be used to directly form the "weighting functions" for the individual parameters θ_{ij}.

An estimate of the position dependent quantities in equations 7.5 – 7.6 is given by:

$$\hat{\mathcal{F}}_1(\mathbf{x}_d) = \sum_{i,j,k=0}^{M} \hat{\Theta}_{1_{ijk}} \varphi_{ijk}(\mathbf{x}_d) \tag{7.19}$$

$$\hat{\mathcal{F}}_2(\mathbf{x}_d) = \sum_{i,j,k=0}^{M} \hat{\Theta}_{2_{ijk}} \varphi_{ijk}(\mathbf{x}_d) \tag{7.20}$$

and the estimate of the friction function from equation 7.7 is the same as that for the FCAL:

$$\hat{\mathbf{F}}(\dot{\mathbf{x}}_d) = \sum_{i=0}^{N} \hat{\Psi}_i \phi_i(\dot{\mathbf{x}}_d) \tag{7.21}$$

Therefore, the estimate of the total desired feedforward term is:

$$\hat{\mathbf{w}}_d = \hat{\mathcal{F}}_1(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \overline{\mathbf{C}}(\hat{\mathcal{F}}_1(\mathbf{x}_d), \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \hat{\mathcal{F}}_2(\mathbf{x}_d) + \hat{\mathbf{F}}(\dot{\mathbf{x}}_d) \tag{7.22}$$

For compactness, we will reparameterize the previous equation as we did for the FCAL:

$$\hat{\mathbf{w}}_d = \mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\hat{\mathbf{p}}_c \tag{7.23}$$

where $\hat{\mathbf{p}}_c$ contains the shape function coefficient estimates $\hat{\Theta}_{ijk}$ and $\hat{\Psi}_i$. For development of the error equations and stability proof we will also define:

$$\mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{p}_c = \overline{\mathcal{F}}_1(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \overline{\mathbf{C}}(\overline{\mathcal{F}}_1(\mathbf{x}_d), \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \overline{\mathcal{F}}_2(\mathbf{x}_d) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d) \tag{7.24}$$

where, similar to equation 7.23, $\mathbf{p}_c$ contains Θ_{ijk} and Ψ_i. Again, we will use the following standard update law:

$$\frac{d}{dt}\hat{\mathbf{p}}_c = -\mathbf{K}_a \mathbf{W}_c{}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_v \qquad \mathbf{K}_a > 0 \tag{7.25}$$

Applying the control law and update law of equations 7.10 and 7.25 to equation 7.1, we obtain the following error dynamics:

$$\begin{aligned} \mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{e}_v &= -\mathbf{K}_v\mathbf{e}_v - \mathbf{K}_p\mathbf{e} - \mathbf{q}_n(\mathbf{e}_v, \mathbf{e}) - \overline{\mathbf{C}}(\mathbf{M}_j(\mathbf{x}_j), \dot{\mathbf{x}}_j)\mathbf{e}_v \\ &\quad -\Delta\mathbf{w}_c(\mathbf{e}_v, \mathbf{e}) + \mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\tilde{\mathbf{p}}_c + \overline{\mathbf{d}} \end{aligned} \tag{7.26}$$

$$\frac{d}{dt}\mathbf{e} = \mathbf{e}_v - \lambda\mathbf{e} \tag{7.27}$$

$$\frac{d}{dt}\tilde{\mathbf{p}}_c = -\mathbf{K}_a \mathbf{W}_c{}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_v \tag{7.28}$$

where $\mathbf{q}_n(\mathbf{e}_v, \mathbf{e})$ is the same as that defined for the FCAL, and $\tilde{\mathbf{p}}_c$ is defined as:

$$\tilde{\mathbf{p}}_c = \hat{\mathbf{p}}_c - \mathbf{p}_c \tag{7.29}$$

$\overline{\mathbf{d}}$ is a lumped, bounded disturbance which includes the disturbances contained in $\mathbf{d}$ from equation 7.1 as well as any additional disturbance due to using the approximations of equations 7.5–7.7:

$$\overline{\mathbf{d}} = \mathbf{d} + \mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{p}_c - \left[\mathbf{M}_j(\mathbf{x}_d)\ddot{\mathbf{x}}_d + \overline{\mathbf{C}}(\mathbf{M}_j(\mathbf{x}_d), \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d)\right] \tag{7.30}$$

$\Delta\mathbf{w}_c(\mathbf{e}_v, \mathbf{e})$ is once again a state dependent disturbance due to using the *desired* trajectory quantities instead of their *true* counterparts, and is defined by:

$$\begin{aligned} \Delta\mathbf{w}_c \;=\; & \left[\mathbf{M}_j(\mathbf{x}_j)\frac{d}{dt}\mathbf{v}_j + \overline{\mathbf{C}}(\mathbf{M}_j(\mathbf{x}_j), \dot{\mathbf{x}}_j)\mathbf{v}_j + \mathbf{g}(\mathbf{x}_j) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_j)\right] \\ & - \left[\mathbf{M}_j(\mathbf{x}_d)\frac{d}{dt}\dot{\mathbf{x}}_d + \overline{\mathbf{C}}(\mathbf{M}_j(\mathbf{x}_d), \dot{\mathbf{x}}_d)\dot{\mathbf{x}}_d + \mathbf{g}(\mathbf{x}_d) + \overline{\mathbf{F}}(\dot{\mathbf{x}}_d)\right] \end{aligned} \tag{7.31}$$

In a direct parallel with the DCAL and FCAL, we present the following stability theorem for the CLL:

Theorem 7.2.1 *Under the assumption of a persistently exciting* $\mathbf{W}_c(t)$ *(see equation 4.36), the results of theorems 4.3.1–4.3.3 hold for the error system described by equations 7.26–7.28.*

Proof: The proof of the theorem once again follows directly from the proofs of theorems 4.3.1–4.3.3. □

7.3 CLL Implementation Results

The CLL was implemented on the IBM 7545 robot for joint space control. The performance of the DCAL and FCAL schemes will also be presented in this chapter for relative performance evaluation. In comparison to the FCAL, the CLL demonstrated approximately the same convergence rate and slightly better steady state tracking. Recall that the only

difference between these two schemes is the way in which the dynamic portion (the part of the equation of motion *not* associated with friction) of the feedforward signal is generated. Because the CLL's representation of this portion of the feedforward term is more general than that used for the FCAL, it is capable of canceling additional dynamic effects. This slightly better performance is achieved with the added advantage of *not* having to generate the specific equations of motion as is required with the DCAL and FCAL.

The same joint space trajectory as shown in figure 4.1 was used for all three controllers. The first order triangular shape function presented in example 7.2.1 was used in conjunction with the CLL's position dependent feedforward term, and the piecewise linear shape function of example 6.2.1 was used for friction estimation in both the FCAL and CLL.

Figures 7.2, 7.3, and 7.4 show position error for the DCAL, FCAL, and CLL respectively over six cycles of the desired trajectory for the first two (revolute) axes. Because the performances of the FCAL and CLL are very close and very near the resolution capabilities of the IBM 7545, the DCAL results are presented for relative comparison. Note that the CLL converges slightly faster than the FCAL, but the steady state error for the FCAL is slightly better than that of the CLL. Also note that all of the position error plots are on the same ordinate scale for easy comparison.

Figures 7.5 and 7.6 show the RMS (root mean square) error over the *first* and *last* cycles of the three controllers. These figures represent an average of the RMS error for the first two axes. Both the FCAL and CLL show similar performance during these two stages of trajectory execution.

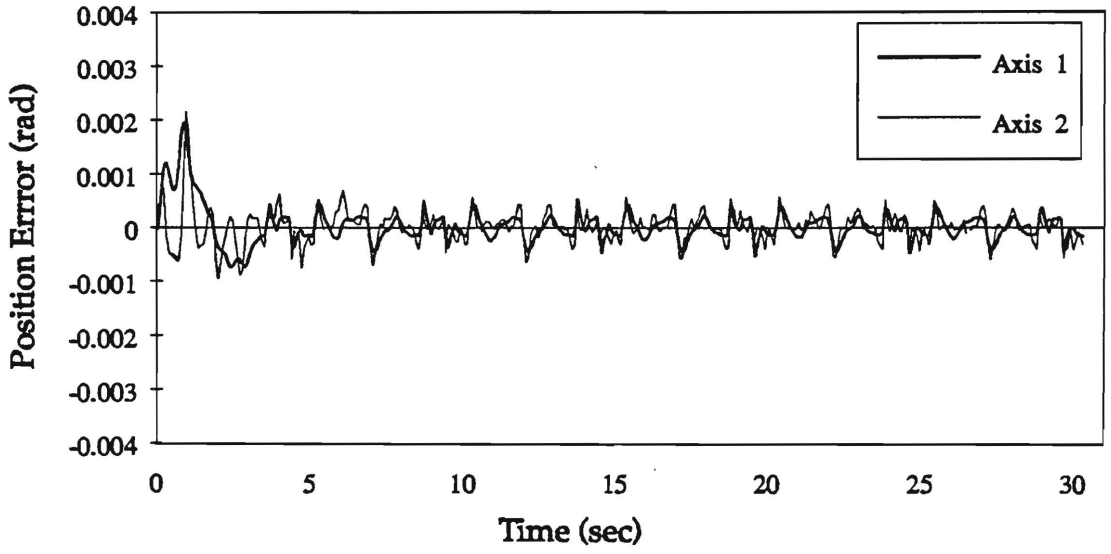

Figure 7.2: DCAL Joint Position Error

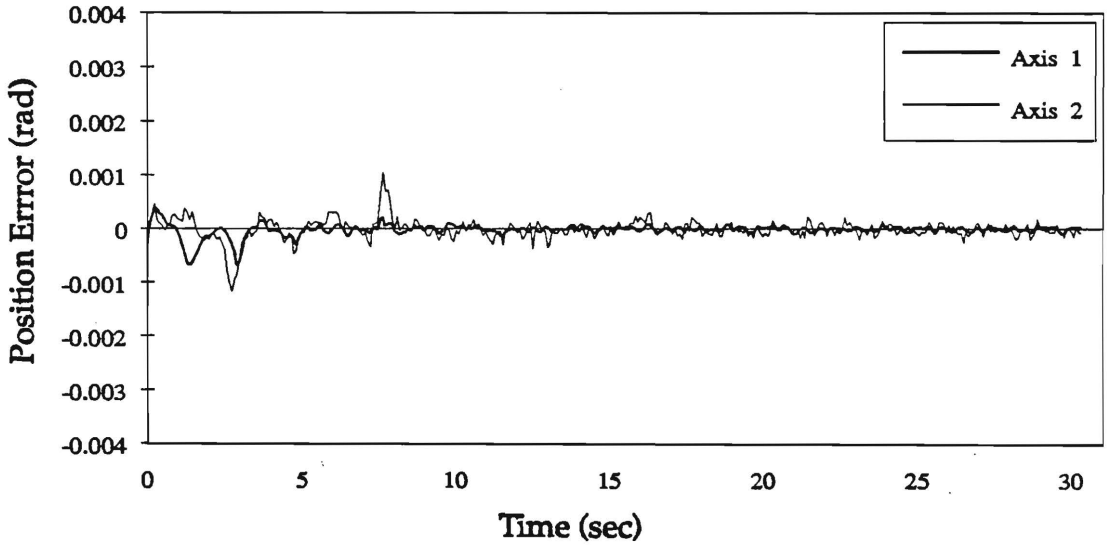

Figure 7.3: FCAL Joint Position Error

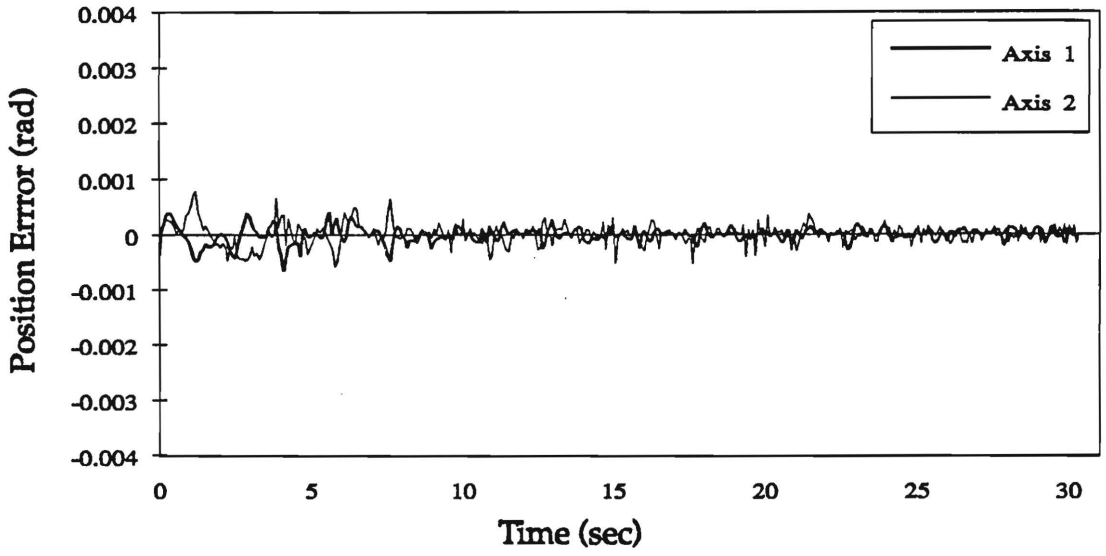

Figure 7.4: CLL Joint Position Error

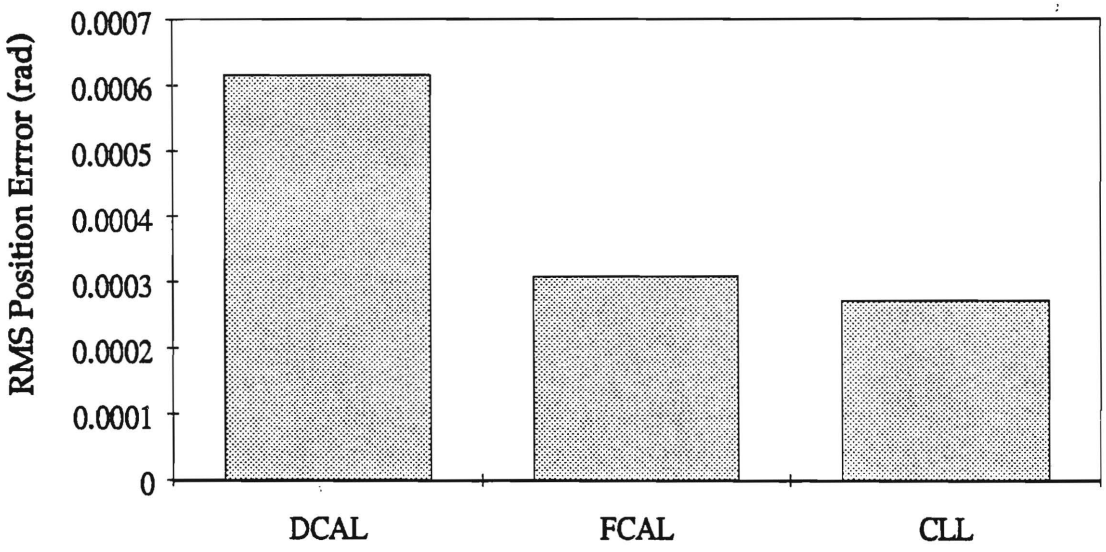

Figure 7.5: First Cycle RMS Joint Position Error

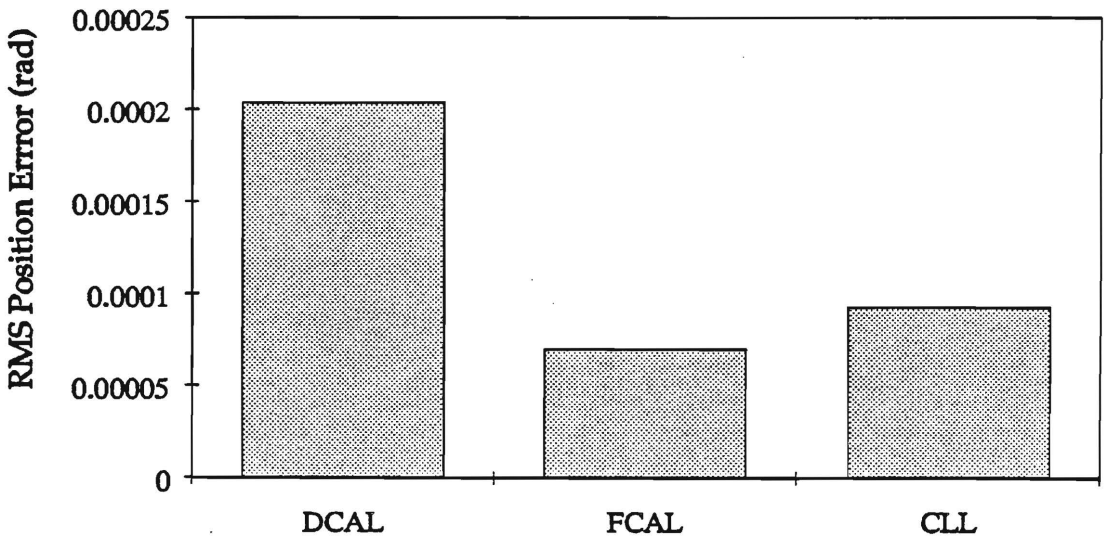

Figure 7.6: Steady State Cycle RMS Joint Position Error

The FCAL, however, would have the advantage if the trajectory were to suddenly change and place the manipulator in a configuration in which it has never been. At this point the CLL would have to converge its position dependent surface from initial conditions in order to generate the required feedforward estimate. Since the robot's path must be continuous, however, previously learned portions of the position dependent hypersurface serve as initial conditions for new areas. After the robot has executed trajectories in many different areas of its configuration space, no new convergence time will be required.

Overall, very similar performance was observed for the FCAL and CLL. The CLL, however, does not require derivation of the specific equation of motion for the system. Moreover, the CLL demonstrates a method of approximating the equation of motion of a system whose dynamics may be unknown or highly complex without making restrictions on the desired trajectory.

7.4 CLL – DCLL Parallel Implementation

Although the performance of the CLL is excellent, it does not exceed that of the DCLL when the desired trajectory is periodic. This is due to the DCLL's ability to identify and reject periodic disturbances common in mechanical systems. Therefore, it would be desirable to use the DCLL during execution of a periodic trajectory, and to use the CLL during a nonperiodic trajectory. Moreover, it would be desirable to update the learning parameters associated with the scheme *not* in control by attempting to match the feedforward input generated by the current controller. In this section we will address a method of accomplishing this parallel update.

If information is available that describes the periodicity of the desired task, then either the DCLL or CLL may be easily "switched on" to operate in their corresponding trajectory domains. If the task is periodic, then the DCLL will have control of the system and will be calculating a feedforward signal based on learned coefficients of the shape functions needed to form this signal (see equation 5.12). During this time, the CLL's feedforward estimate may be updated to match that of the (higher performance) DCLL. The *new* parameter update equation for the CLL, similar to equation 7.25, would be:

$$\frac{d}{dt}\hat{\mathbf{p}}_c = -\mathbf{K}_{ap}\mathbf{W}_c{}^T(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\mathbf{e}_q \qquad \mathbf{K}_{ap} > 0 \tag{7.32}$$

$\mathbf{e}_q$ is the feedforward torque error defined as

$$\mathbf{e}_q = \mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\hat{\mathbf{p}}_c - \hat{\mathbf{w}}_d \tag{7.33}$$

where $\hat{\mathbf{w}}_d$ is the feedforward signal generated by the DCLL (see equation 5.12).

Another advantage of a parallel implementation of the DCLL and CLL controllers would occur during the transition from a nonperiodic to a periodic task. During the first cycle of execution of the DCLL, each shape function coefficient could be initialized to reflect the CLL's calculated feedforward signal based on the same desired trajectory values. By equating the feedforward signals of both controllers we have

$$\sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) = \mathbf{W}_c(\mathbf{x}_d, \dot{\mathbf{x}}_d, \ddot{\mathbf{x}}_d)\hat{\mathbf{p}}_c \tag{7.34}$$

where the left hand side of the equation is the feedforward term for the DCLL, and the right hand side is the feedforward term for the CLL. If N of these equations are generated for each of N points in the desired trajectory, each parameter vector $\hat{\Theta}_i$ may be solved for uniquely since the shape functions ϕ_i are linearly independent. This reduces to a simple solution if the piecewise linear shape function described in part (b) of example 5.2.1 is used. In this case $\hat{\Theta}_i$ is exactly equal to the calculated feedforward torque from the CLL at the beginning (or end) of each linear segment.

7.5 CLL – DCLL Parallel Implementation Results

We will examine the performance of a parallel implementation of the CLL and DCLL control schemes as described in the previous section. First, the DCLL will be placed in control of the manipulator while executing the previous periodic trajectory for 6 cycles. During this time the CLL will use equation 7.32 to update its parameters from initial conditions. At this point the DCLL will be switched off, the CLL will take control, and its first cycle performance will be compared to that of section 7.3. In the second situation, the CLL will be placed in control of the robot while executing the same periodic trajectory. After 2 cycles of execution, the DCLL using a piecewise linear approximation will be switched on, and its parameters will be initialized using the CLL's approximation of the required feedforward torque (as in equation 7.34).

Figures 7.7 and 7.8 show the joint position error time history of the two parallel implementation schemes. In figure 7.7, the CLL is switched on after approximately 30 seconds (6 cycles) of DCLL execution, and in figure 7.8 the DCLL is switched on after 10 seconds (2 cycles) of CLL execution. Note that both plots are on the same ordinate scale. The "first" cycle error for these two cases is represented in figures 7.9 and 7.10. Each figure shows the average of the RMS joint position error for the first two axes during the first cycle of the indicated controller with and without parallel initialization. The CLL achieves a 61% decrease in position error during this cycle, while the DCLL's error is reduced by 88%! This demonstrates that the repetitive controller may be used anytime a periodic task

is to be executed without the need for several convergence cycles, and that the nonrepetitive controller benefits from parallel initialization with the DCLL's computed feedforward signal.

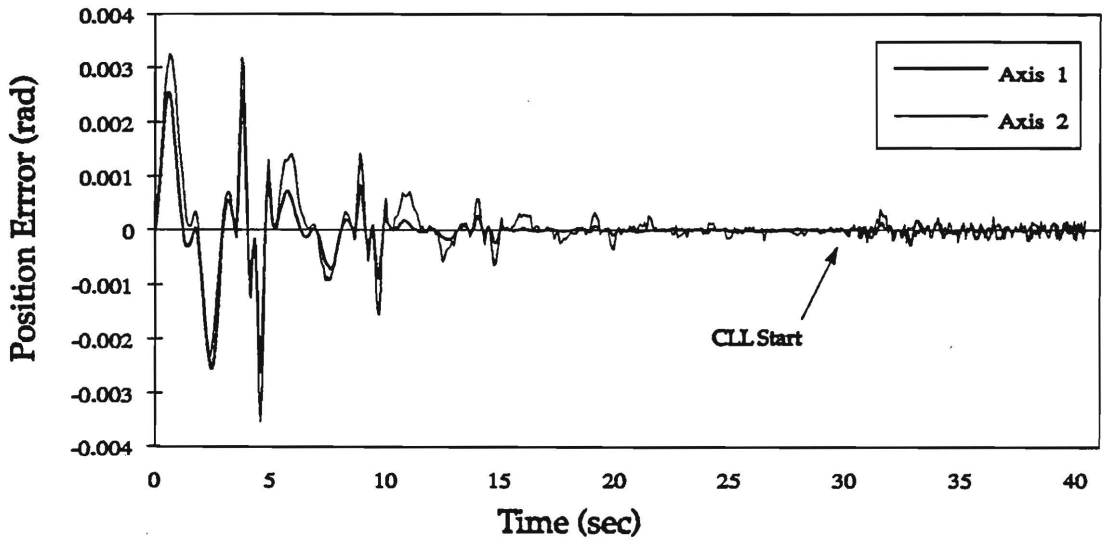

Figure 7.7: DCLL – CLL Parallel Implementation Time History

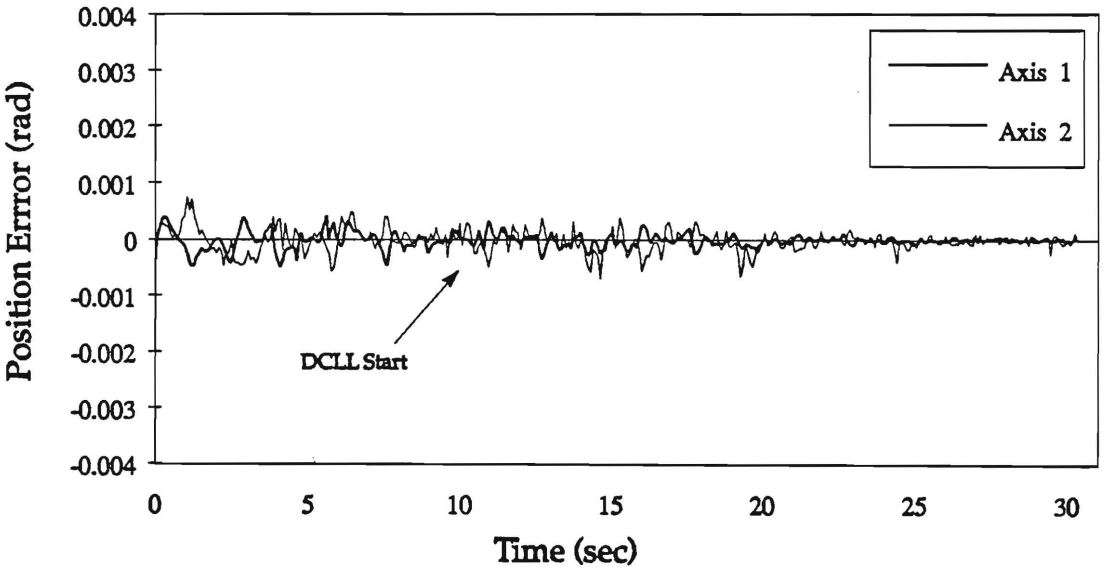

Figure 7.8: CLL – DCLL Parallel Implementation Time History

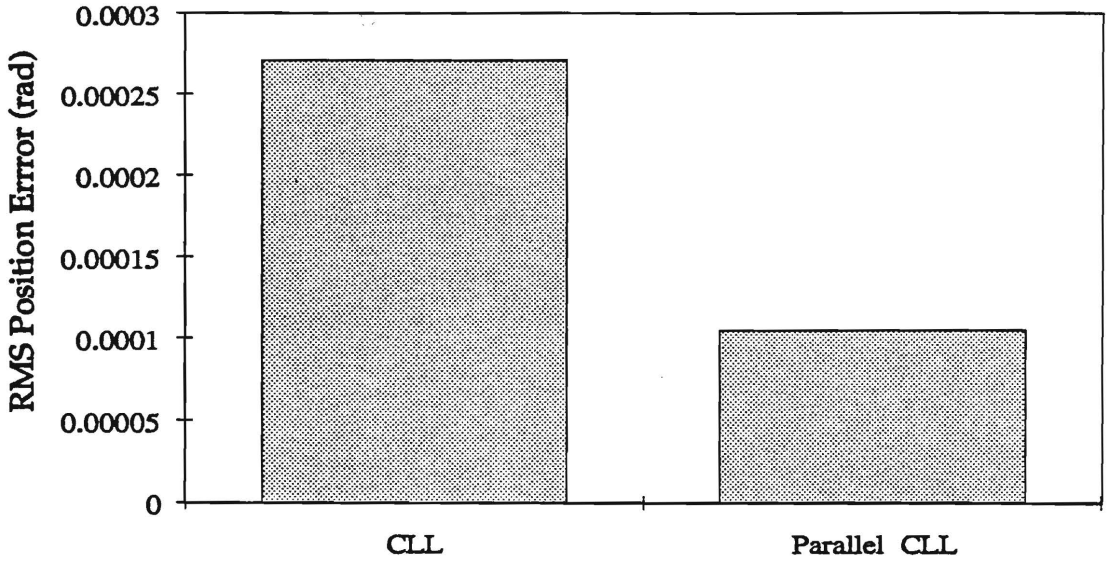

Figure 7.9: Parallel Update CLL First Cycle Joint Position Error

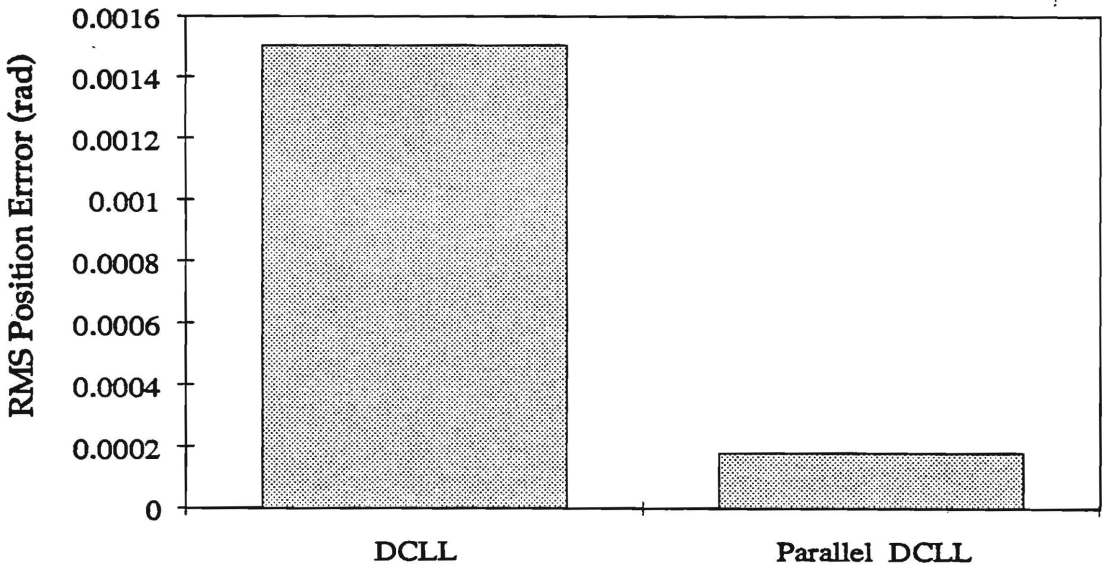

Figure 7.10: Parallel Update DCLL First Cycle Joint Position Error

Chapter 8

Learning Force Control

8.1 Introduction

Until this point, we have focused our attention on *position* control of mechanical manipulators. However, the concept of learning control is also easily extended to *force* control of robots. By utilizing an end–effector mounted force sensor, the robot's normal contact force with an object may be monitored and made to track a given "force trajectory."

Force control on a multi–DOF manipulator, whether it be a numerically controlled machine tool, robotic manipulator, or any other mechanical system, usually involves force/torque control along only *one* axis. Position control must be maintained in the other direction(s), and interaction between the two modes of control must be cooperative. This type of "dual" control has been labeled *hybrid control* as a matter of consensus in the robotics literature. This is an appropriate term since the control system must be made up of two distinctly different parts while exhibiting the properties of both in a highly blended fashion.

We will address the formulation of a *repetitive* hybrid force control scheme in this chapter. Much of the problem in applying advanced adaptive position control schemes to force control is due to the fundamentally *Cartesian* nature of the task. However, recall that the DCLL was easily applied to direct execution of a Cartesian task. This controller will form the "kernel" of our hybrid force control algorithm. The overall topology of the integrated

system may be outlined as follows:

1. Cartesian position control will be applied in the directions tangential to an interaction surface.
2. Force control will be applied in the direction normal to the surface.
3. The rotational degree of freedom (the "roll" axis) will be used to maintain normal contact of the end–effector.

Therefore, simultaneous repetitive learning control will be applied in three different regimes – Cartesian position control, force control, and trajectory generation. We will refer to this hybrid controller as the Hybrid Learning Law (HLL). The performance of the HLL will be compared to that of a PID (Proportional–Integral–Derivative) system typical in today's industrial controllers.

8.2 HLL Force Control Development

In this section we will apply the DCCLL Cartesian *position* controller presented in section 5.3 to *force* control of the end–effector of a robot. The following development will be based on force control of a *single* degree of freedom of a multiple DOF manipulator. This is done for simplicity of development and is actually the way in which the hybrid force control scheme will be implemented. The manipulator will be assumed to have a 6 DOF wrist force/torque sensor.

Much of the work has already been done in extending the DCCLL to force control since Cartesian position control is closely related to this problem. We assume that the interaction of the robot end–effector with an object possesses a *bounded* translational stiffness in some direction z:

$$F_z = k(z)\,\Delta z \tag{8.1}$$

where $k(z)$ is a possibly nonlinear stiffness coefficient, and F_z is the force generated at the robot wrist in the direction z due to some deformation of the object Δz. Since we assume

that $k(z)$ is bounded, we may write

$$\sup_{z} \|k(z)\| < \delta \qquad \text{for some } \delta > 0 \tag{8.2}$$

This assumption requires that there be some inherent flexibility in either the robot, the contact surface, or both. If we exclude singular configurations of the manipulator by requiring the condition set forth in equation 3.13, then there is certainly always some compliance in the robot because of limited torque and the feedback control law that surrounds the actuators. This validates the assumption of equation 8.2.

To formulate the force control law let us first define the *force* errors similar to their position counterparts. Define the Cartesian force tracking error, e_f, by:

$$e_f = F_{\text{tip}} - F_d \tag{8.3}$$

where F_{tip} is the sensed force at the end–effector in some direction z, and F_d is the *desired* force at the tip in this direction. Define the *reference force derivative error*, e_{f_v} as:

$$e_{f_v} = \dot{e}_f + \lambda_f e_f \tag{8.4}$$

We may write the force control law as:

$$q = -K_p^f e_f - K_v^f e_{f_v} - K_{\text{damp}}(\dot{z} - \dot{z}_d) - q_{fn}(e_{f_v}, e_f) + \hat{w}_{fd}(t) \tag{8.5}$$

where K_{damp} is a constant damping coefficient, and $\dot{z}$ describes the velocity of the end–effector in the z tool direction. $\dot{z}_d$ is the desired velocity of the end–effector in the tool direction, similar to the time derivative of the desired force, and is approximately zero due to the stiffness of the system in this direction. This term, along with the $K_v^f e_{f_v}$ term, add enough artificial damping to allow higher proportional and learning gains. q is the imaginary end–effector Cartesian force in the z tool direction and may be mapped to the "world" coordinate system using a simple rotation transformation. If $\mathbf{z} = [\ 0 \quad 0 \quad z\ ]^T$ then we may write (see [3], [10]):

$$\Rightarrow \qquad \Delta \mathbf{x}_c = \mathbf{R}^T(\mathbf{x}_j)\mathbf{z} \qquad \Rightarrow \tag{8.6}$$

where $\mathbf{R}^T(\mathbf{x}_j)$ is the rotation transformation to the "world" coordinate vector $\Delta\mathbf{x}_c$. The actual input to the manipulator actuators $\mathbf{q}_j$ is written as:

$$\mathbf{q}_j = \mathbf{J}^T(\mathbf{x}_j)\mathbf{R}(\mathbf{x}_j)\mathbf{q} \tag{8.7}$$

where $\mathbf{q} = [\ 0 \quad 0 \quad q\ ]^T$.

The other terms in equation 8.5 are scalar force analogs of the position quantities given in section 5.3, but are listed here for completeness. q_{fn} is the nonlinear force feedback term given by

$$q_{fn}(e_{f_v}, e_f) = \sigma_{nf}\,|e_f|^2\, e_{f_v} \qquad \sigma_{nf} > 0 \tag{8.8}$$

An estimate of the feedforward term is given by

$$\hat{w}_{fd}(t) = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{8.9}$$

with the coefficient update law

$$\frac{d}{dt}\hat{\Theta}_i = -K_{l_i}^f \phi_i(t) e_{f_v} \qquad K_{l_i}^f > 0 \tag{8.10}$$

Although similar in form to the Cartesian position control version, the force control law exhibits very different behavior. According to experimental observations, a large "integral" and derivative along with a small proportional control is desirable, where the integrator action is provided by the update law of equation 8.10. For position control, a large proportional and light derivative and "integral" gains provide the best all–around performance.

8.3 HLL Position and Force Control Integration

In this section we will integrate the repetitive Cartesian *position* controller developed in section 5.3 and the force controller presented in the previous section into a hybrid force/position algorithm. This hybrid control law will be applicable to tracking a desired position trajectory tangential to a contact surface, and a desired force trajectory normal to the surface.

It will be assumed that the direction of the outward normal of the surface is known at any point although this assumption will be relaxed in the next section.

If $\mathbf{x}_{c_{\text{tool}}}$ is a vector describing the manipulator end–effector's current position with respect to the "surface" or "tool" Cartesian coordinate system, then there is a mapping to a vector $\mathbf{x}_{c_{\text{normal}}}$ describing the surface's outward normal at this point since this direction is assumed to be known:

$$\mathbf{x}_{c_{\text{normal}}} = \eta(\mathbf{x}_{c_{\text{tool}}}) \tag{8.11}$$

Note that $\mathbf{x}_{c_{\text{tool}}}$ describes the actual *position* of the manipulator's end–effector in the surface frame coordinate system, while $\mathbf{x}_{c_{\text{normal}}}$ describes a vector at this point which is *not* a function of the end–effector's *orientation.* We may define the rotation transformation between some "world" or "absolute" Cartesian coordinates and the surface frame coordinate system as $\mathbf{R}(\mathbf{x}_{c_{\text{normal}}})$:

$$\Delta\mathbf{x}_{c_{\text{tool}}} = \mathbf{R}(\mathbf{x}_{c_{\text{normal}}})\Delta\mathbf{x}_{c_{\text{abs}}} \tag{8.12}$$

where $\Delta\mathbf{x}_{c_{\text{tool}}}$ and $\Delta\mathbf{x}_{c_{\text{abs}}}$ are 3 by 1 translation vectors written in the "tool" and "world" coordinate systems, respectively. The end–effector also possesses an orientation which will be described by the 3 by 1 vector $\mathbf{x}_{c_{\text{rot}}}$. Combining $\mathbf{x}_{c_{\text{tool}}}$ and $\mathbf{x}_{c_{\text{rot}}}$ gives the total "tool" or "surface" frame position and orientation vector $\mathbf{x}_s$:

$$\mathbf{x}_s = \begin{bmatrix} \mathbf{x}_{c_{\text{tool}}} \\ \mathbf{x}_{c_{\text{rot}}} \end{bmatrix} \tag{8.13}$$

These *actual* quantities have corresponding *desired* counterparts $\mathbf{x}_{d_{\text{tool}}}$ and $\mathbf{x}_{d_{\text{rot}}}$. The 3 by 1 desired rotation vector, $\mathbf{x}_{d_{\text{rot}}}$, which forms the orientation portion of the total desired position trajectory vector, $\mathbf{x}_d$, is given by:

$$\mathbf{x}_{d_{\text{rot}}}(t) = \underline{\mathbf{D}}\mathbf{x}_d(t) \tag{8.14}$$

$$\underline{\mathbf{D}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.15}$$

In a similar fashion, we may also define the 3 by 1 desired translation vector $\mathbf{x}_{d_{\text{tool}}}$ as:

$$\mathbf{x}_{d_{\text{tool}}}(t) = \overline{\mathbf{D}}\mathbf{x}_d(t) \tag{8.16}$$

$$\overline{\mathbf{D}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{8.17}$$

It will be assumed that the position and force trajectories are defined in terms of the surface coordinates, and that all orientation vectors ($\mathbf{x}_{c_{\text{normal}}}$, $\mathbf{x}_{c_{\text{rot}}}$, $\mathbf{x}_{d_{\text{rot}}}$) are defined in the fixed "world" coordinate system. The control law will act to always keep one of the principal axes of the tool frame aligned with the desired outward normal of the surface. We may assume this without loss of generality since the choice of the tool frame is arbitrary, and we will designate this normal direction as the z axis. The other two principle axes of the tool frame describe the plane in which tangential motion is to take place, and we will call these axes x and y. The first three elements of $\mathbf{x}_d$ contain the desired x, y, and z coordinates, respectively, and the last three elements contain orientation information in accordance with equations 8.16 and 8.14.

As the end–effector moves over the surface, the tool frame (and the end–effector) rotates to maintain proper alignment. In this way, tracking a complex surface is reduced to specifying a trajectory along the surface as well as describing the outward normal to the surface at any point. This may sound complicated at first, but if the trajectory can be easily described in terms of the surface contour it yields a simpler description. For example, if an airplane wing is to be checked with some type of rolling contact ultrasonic sensor, the position trajectory specified would simply be a number of straight lines over the surface contour. Of course, the outward normal to the surface must be specified, but this restriction will be relaxed in the next section.

We may keep track of the current tool frame tangential position relative to an arbitrary starting point by using the following:

$$\Delta \mathbf{x}_{c_{\text{tool}}} = \int_0^t \mathbf{R}(\mathbf{x}_{c_{\text{normal}}}) \dot{\mathbf{x}}_{c_{\text{abs}}} dt \tag{8.18}$$

This integration may be performed numerically on–line while the control scheme is operating without the need to use or compute $\dot{\mathbf{x}}_{c_{\text{abs}}}$ (see section 8.5.2 for details). Note that although

the vector $\Delta \mathbf{x}_{c_{\text{tool}}}$ contains all three Cartesian tool coordinates x, y, and z, only the first two are meaningful in the position control topology. The z coordinate, however, *will* be used in the stability analysis of the force control portion. The position tracking error vector analogous to equation 5.20 is

$$\mathbf{e}_{\text{tool}} = \Delta \mathbf{x}_{c_{\text{tool}}} - \mathbf{x}_{d_{\text{tool}}} \tag{8.19}$$

The velocity vector in terms of the tool frame is simply

$$\dot{\mathbf{x}}_{c_{\text{tool}}} = \mathbf{R}(\mathbf{x}_{c_{\text{normal}}})\dot{\mathbf{x}}_{c_{\text{abs}}} \tag{8.20}$$

and the time derivative of $\mathbf{e}_{\text{tool}}$ is written as

$$\dot{\mathbf{e}}_{\text{tool}} = \dot{\mathbf{x}}_{c_{\text{tool}}} - \dot{\mathbf{x}}_{d_{\text{tool}}} \tag{8.21}$$

We may then write the reference velocity position error analogous to equation 5.22 for tool frame trajectory description as

$$\mathbf{e}_{v_{\text{tool}}} = \dot{\mathbf{e}}_{\text{tool}} + \lambda_c \mathbf{e}_{\text{tool}} \tag{8.22}$$

The *position* control law for the hybrid scheme is written as

$$\mathbf{F}_{p_{\text{tool}}} = -\mathbf{K}_p \mathbf{e}_{\text{tool}} - \mathbf{K}_v \mathbf{e}_{v_{\text{tool}}} - \mathbf{q}_n(\mathbf{e}_{v_{\text{tool}}}, \mathbf{e}_{\text{tool}}) + \hat{\mathbf{w}}_{pd}(t) \tag{8.23}$$

where all of the terms are similar to those described for equation 5.24 except that $\mathbf{F}_{p_{\text{tool}}}$ is a 3 by 1 vector of only the imaginary end–effector *forces*. The remaining *torque* elements of the input vector are obtained from an orientation control law which is written in "world" coordinates, and is exactly the same as the bottom 3 elements of the 6 by 1 vector $\mathbf{q}$ given in equation 5.24:

$$\mathbf{T}_{r_{\text{abs}}} = -\mathbf{K}_p \mathbf{e}_r - \mathbf{K}_v \mathbf{e}_{r_v} - \mathbf{q}_n(\mathbf{e}_{r_v}, \mathbf{e}_r) + \hat{\mathbf{w}}_{rd}(t) \tag{8.24}$$

We restate the scalar *force* control law here in terms of $F_{f_{\text{tool}}}$ for clarity:

$$F_{f_{\text{tool}}} = -K_p^f e_f - K_v^f e_{f_v} - K_{\text{damp}}(\dot{z} - \dot{z}_d) - q_{fn}(e_{f_v}, e_f) + \hat{w}_{fd}(t) \tag{8.25}$$

Recall that all of the errors for the force control law are already in terms of a tool frame. Specifically, the direction z is the same as the contact surface normal, and $\dot{z}$ is the same

as the third component of $\dot{\mathbf{x}}_{c_{\text{tool}}}$. We may now write the control input to the manipulator actuators, $\mathbf{q}_j$, for the HLL control system:

$$\mathbf{F}_{\text{abs}} = \mathbf{R}^T(\mathbf{x}_{c_{\text{normal}}})(\mathbf{A}\mathbf{F}_{p_{\text{tool}}} + \mathbf{B}F_{f_{\text{tool}}}) \tag{8.26}$$

$$\mathbf{T}_{\text{abs}} = \mathbf{T}_{r_{\text{abs}}} \tag{8.27}$$

$$\mathbf{q}_j = \mathbf{J}^T(\mathbf{x}_j)\begin{bmatrix} \mathbf{F}_{\text{abs}} \\ \mathbf{T}_{\text{abs}} \end{bmatrix} \tag{8.28}$$

where $\mathbf{A} = \text{diag}(\ 1 \quad 1 \quad 0\ )$ and $\mathbf{B} = [\ 0 \quad 0 \quad 1\ ]^T$.

To analyze the stability of the proposed hybrid learning scheme, we will write the dynamic equations of motion of the manipulator in the *surface* or *tool* frame of the manipulator. Note that this frame is attached to the contact surface as opposed to the manipulator end–effector. We will assume that the manipulator maintains contact with the surface during execution of the desired task. Similar to definition 3.3.1, we may write the mapping from the joint space to the tool or surface space as $\mathbf{x}_s = \overline{\zeta}(\mathbf{x}_j)$. The Jacobian for this mapping is therefore

$$\mathbf{J}_s(\mathbf{x}_j) \equiv \frac{d\overline{\zeta}(\mathbf{x}_j)}{d\mathbf{x}_j} \tag{8.29}$$

This new Jacobian, $\mathbf{J}_s(\mathbf{x}_j)$, may be used to transform the joint space mass matrix as in equation 3.12. The surface frame equation of motion for the manipulator is written as

$$\begin{aligned} \frac{d}{dt}\mathbf{x}_s &= \dot{\mathbf{x}}_s \\ \mathbf{M}_s(\mathbf{x}_s)\frac{d}{dt}\dot{\mathbf{x}}_s + \mathbf{C}(\mathbf{x}_s, \dot{\mathbf{x}}_s)\dot{\mathbf{x}}_s + \mathbf{g}_s(\mathbf{x}_s) + \mathbf{K}\mathbf{x}_s &= \mathbf{q}_s + \mathbf{d} \end{aligned} \tag{8.30}$$

where we assume that the stiffness matrix $\mathbf{K}$ has the form

$$\mathbf{K} = \text{diag}\begin{pmatrix} 0 & 0 & k_z & k_{rx} & k_{ry} & 0 \end{pmatrix} \tag{8.31}$$

k_z is the linear stiffness in the z (normal) direction, and k_{rx} and k_{ry} are the rotational stiffnesses about the x and y axes (surface tangential directions). $\mathbf{q}_s$ is the imaginary input force at the end–effector described in surface frame coordinates

$$\mathbf{q}_s = \begin{bmatrix} \mathbf{A}\mathbf{F}_{p_{\text{tool}}} + \mathbf{B}F_{f_{\text{tool}}} \\ \mathbf{R}(\eta(\overline{\mathbf{D}}\mathbf{x}_s))\mathbf{T}_{r_{\text{abs}}} \end{bmatrix} \tag{8.32}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the same as in equation 8.26, and $\mathbf{R}(\eta(\overline{\mathbf{D}}\mathbf{x}_s))$ yields the same rotation transformation as equation 8.12.

For simplicity in the stability analysis, we may set the stabilizing gain K_{damp} to zero and make the following substitutions in the force control law of equation 8.25:

$$e_f = k_z e_z \tag{8.33}$$

$$e_{f_v} = k_z e_{z_v} \tag{8.34}$$

where

$$e_z = z - z_d \tag{8.35}$$

$$e_{z_v} = \dot{z} - v_z = \frac{d}{dt} e_z + \lambda_f e_z \tag{8.36}$$

and z_d is proportional to the desired force F_d. Applying the hybrid control law from equation 8.32 with these modifications to equation 8.30, we arrive at the following error dynamic system:

$$\begin{aligned} \mathbf{M}_s(\mathbf{x}_s)\frac{d}{dt}\mathbf{e}_{s_v} &= -\mathbf{K}_{sv}\mathbf{e}_{s_v} - \mathbf{K}_{sp}\mathbf{e}_s - \mathbf{q}_n - \mathbf{C}(\mathbf{x}_s, \dot{\mathbf{x}}_s)\mathbf{e}_{s_v} \\ &\quad -\Delta\mathbf{w}(\mathbf{e}_{s_v}, \mathbf{e}_s) + \mathbf{W}_s\tilde{\Theta} + \overline{\mathbf{d}} \end{aligned} \tag{8.37}$$

$$\frac{d}{dt}\mathbf{e}_s = \mathbf{e}_{s_v} - \lambda_s \mathbf{e}_s \tag{8.38}$$

$$\frac{d}{dt}\tilde{\Theta} = -\mathbf{K}_{sl}\mathbf{W}_s^T\mathbf{e}_{s_v} \tag{8.39}$$

where $\tilde{\Theta}$, $\Delta\mathbf{w}(\mathbf{e}_{s_v}, \mathbf{e}_s)$, and $\overline{\mathbf{d}}$ are similar to the quantities defined for the Cartesian space error system in equations 5.29–5.31. $\mathbf{K}_{sv}$, $\mathbf{K}_{sp}$, λ_s, and $\mathbf{K}_{sl}$ are simply diagonal gain matrices composed of the corresponding gains from the position and force control laws, and $\mathbf{e}_s = \mathbf{x}_s - \mathbf{x}_d$ is the total surface frame error vector.

In an analogous fashion to the pure Cartesian position control law, we state the stability theorem for the HLL.

Theorem 8.3.1 *Due to the natural persistent excitation of the "weighting" matrix $\mathbf{W}_s(t)$ (see equation 5.14), and under the assumption of equation 3.13, there exists a set of control*

gains $\mathbf{K}_{sp}$, $\mathbf{K}_{sv}$, σ_n, *and* σ_{nf} *such that the results of theorems 5.3.1–5.3.2 hold for the error system given by equations 8.37–8.39.*

Proof: The proof of the theorem follows directly from the proofs of theorems 5.3.1–5.3.2 and 4.3.2–4.3.3. □

8.4 Orientation Control

In the last section it was pointed out that the normal to the contact surface must be known at all points to implement the hybrid controller. In this section we will explore a way to relax this restriction so that an ***unknown*** surface may be tracked. Toward this end, we will utilize the wrist force/torque sensor's torque sensing capability to generate a desired orientation trajectory on–line. By examining the torque present about the desired contact point, corrections can be made in the end–effector orientation to maintain normal contact.

Figure 8.1 shows a diagram of the situation which exist when the end–effector is not correctly aligned with the surface. The force control portion of the hybrid scheme maintains a specified normal contact force, and motion tangential to the surface generates a friction force. We may map the tool center point for the force/torque sensor out to the desired contact point using the following transformation

$$\mathbf{T}_{\text{tool}} = \mathbf{T}_{\text{sensed}} + \mathbf{r} \times \mathbf{F}_{\text{sensed}} \tag{8.40}$$

where $\mathbf{r}$ is the vector from the desired contact point to the force/torque sensor center point, $\mathbf{T}_{\text{sensed}}$ and $\mathbf{F}_{\text{sensed}}$ are the sensed torque and force vectors about the sensor center point, and $\mathbf{T}_{\text{tool}}$ is the torque generated about the desired contact point. Note that all of these vectors are naturally written with respect to the tool coordinate frame. The friction force does not generate a torque about the desired contact point because there is no moment arm for it to act through. The normal force, however, ***can*** generate a torque component in $\mathbf{T}_{\text{tool}}$ if it is not collinear with the end–effector center line. When this is detected, the orientation of the end–effector can be changed to realign the normal force vector with the

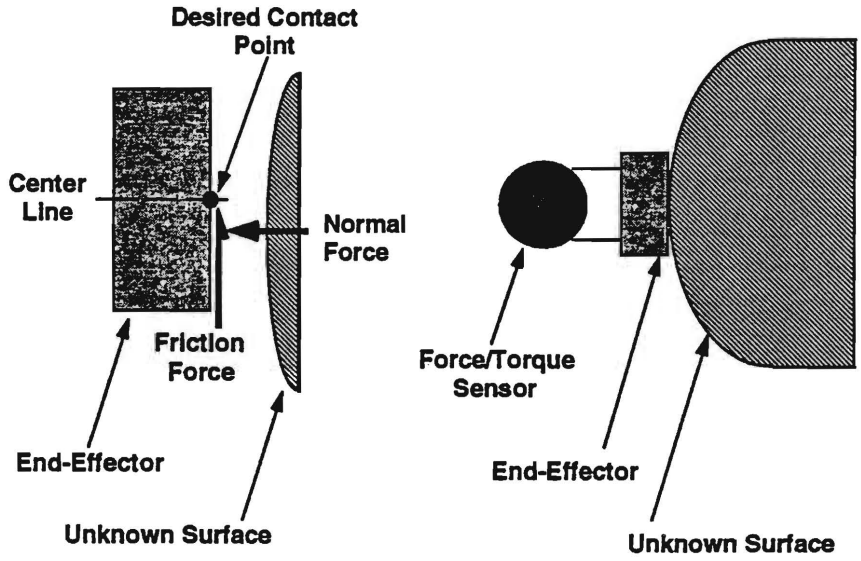

Figure 8.1: Orientation Learning Diagram

center line. Therefore, $\mathbf{T}_{\text{tool}}$ becomes an effective *error* signal for the proposed task of maintaining normal contact to an unknown surface with respect to a desired contact point and center line.

We can define a trajectory update law as

$$\mathbf{x}_{d_{\text{rot}}}(t) = \underline{\mathbf{D}}\mathbf{x}_d(t) + \mathbf{x}_{d_{\text{learn}}}(t) \tag{8.41}$$

where $\mathbf{x}_{d_{\text{rot}}}(t)$ is once again the 3 by 1 vector of desired orientation quantities for the end-effector in terms of some "world" coordinate system. $\mathbf{x}_{d_{\text{learn}}}(t)$ is the learned portion of the orientation trajectory, and the product of $\underline{\mathbf{D}}\mathbf{x}_d(t)$ is the original desired orientation trajectory (see equations 8.14 and 8.15). This term is included in the formulation of a "learned" trajectory to take advantage of any knowledge of the surface that may exist. In this way the learned portion of the trajectory perturbs the known trajectory to compensate for the shape of the true surface.

To use the learned orientation information in the control law of equation 8.26, we must assume that the surface orientation is changing slowly, or, more precisely:

$$\mathbf{x}_{c_{\text{normal}}}(t) \approx \mathbf{x}_{d_{\text{rot}}}(t) \tag{8.42}$$

at some point $\mathbf{x}_{c_{\text{tool}}}(t)$. As the repetitive control law cycles and drives the position, force, and orientation error to zero, this approximation becomes better and better. The form of $\mathbf{x}_{d_{\text{learn}}}(t)$ is similar to that of the feedforward terms for the position and force control laws:

$$\mathbf{x}_{d_{\text{learn}}}(t) = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{8.43}$$

Since no proportional or derivative terms are involved, the trajectory estimation is based on only the integral action of the parameter update law. The error signal is simply the torque generated about the tool center point $\mathbf{T}_{\text{tool}}$:

$$\frac{d}{dt}\hat{\Theta}_i(t) = -\mathbf{K}_{ti}\phi_i(t)\mathbf{R}^T(\mathbf{x}_{d_{\text{rot}}})\mathbf{E}\,\mathbf{T}_{\text{tool}}(t) \qquad \mathbf{K}_{ti} > 0 \tag{8.44}$$

the rotation matrix $\mathbf{R}^T(\mathbf{x}_{d_{\text{rot}}})$ is the inverse (transpose) of the rotation matrix given in equation 8.12, and maps the tool frame torque into the "world" coordinate system. $\mathbf{E}$ is a masking matrix defined as

$$\mathbf{E} = \text{diag}\begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$$

which removes the torque about the tool centerline (surface normal) since making an estimation of this rotation is not defined in this scheme.

Of course, this algorithm is not capable of correctly detecting or correcting all orientation errors. Small concave "cups" or internal corners in the surface may not be negotiated, but convex "bumps" and external corners are usually handled very well. Also, equation 8.44 is only valid for incremental changes in the trajectory, and "large" rates of change of orientation may not be correctly learned. This trajectory generation scheme is only meant to serve as an example of application of the general repetitive learning algorithm, and the details of defining and solving the problems of generalized trajectory determination are an entire field unto themselves. However, this scheme has demonstrated excellent performance with reasonable test cases.

8.5 HLL Implementation Details

In this section we will outline some of the specific methods used to implement the HLL on the IBM 7545 robot. The HLL was implemented in the repetitive learning law form described in the previous sections as well as in a pure PID form for comparison purposes.

8.5.1 HLL Implementation in PID Form

Since a PD feedback loop is already in place within the position and force control laws of equations 5.24 and 8.5, the feedforward term will simply be replaced by the integral component to generate a PID implementation of the HLL. This turns out to be a special case of the selection of the shape function for the feedforward compensation.

Recall the form of the feedforward term and its associated update law from equations 5.25, 5.26, 8.9, 8.10:

$$\hat{\mathbf{w}}_d = \sum_{i=0}^{N} \hat{\Theta}_i \phi_i(t) \tag{8.45}$$

$$\frac{d}{dt}\hat{\Theta}_i(t) = -\mathbf{K}_l \phi_i(t) \mathbf{e}_v(t) \tag{8.46}$$

If we choose $\phi_i(t) = 1$ and $N = 0$, we obtain:

$$\hat{\mathbf{w}}_d = -\mathbf{K}_l \int_0^t \mathbf{e}_v(t) dt \tag{8.47}$$

or, more precisely

$$\hat{\mathbf{w}}_d = -\lambda \mathbf{K}_l \int_0^t \mathbf{e}(t) dt - \mathbf{K}_l \mathbf{e}(t) \tag{8.48}$$

The first term is exactly a classic "Integrator" with a gain of $\lambda \mathbf{K}_l$, and the second term just adds to the proportional control action already present in the control laws. If we choose the nonlinear control gains $\sigma_n = 0$ and $\sigma_{nf} = 0$ in the position and force control laws, respectively, then we are left with a classic PID implementation of the hybrid control scheme developed for the DCLL. This simplification of the DCLL to a pure "integrator" also serves to highlight the integral action of the periodic learning compensation.

The learning trajectory update of equation 8.44 may be modified in a similar fashion to yield a pure "Integrator" by the same selection of $\phi_i(t) = 1$ and $N = 0$. No PD feedback is needed or desired in the case of the trajectory update since all changes in this quantity should happen slowly with respect to the underlying position and force control system.

8.5.2 HLL Implementation Equations

Motion in the *tool* frame of the end–effector is tracked by numerically integrating equation 8.18. If the contact surface normal vector is changing slowly with respect to the feedback update rate (as it must be for any reasonable surface), then we may assume that the rotation matrix $\mathbf{R}(\mathbf{x}_{c_{\text{normal}}})$ is constant over the sample period. Therefore, for the duration of one sample time, t_c, equation 8.18 simplifies to:

$$\Delta\mathbf{x}_{c_{\text{tool}}}(t) \approx \Delta\mathbf{x}_{c_{\text{tool}}}(t - t_c) + \mathbf{R}(\mathbf{x}_{c_{\text{normal}}}) \int_{t-t_c}^{t} \dot{\mathbf{x}}_{c_{\text{abs}}}(t)dt \tag{8.49}$$

or

$$\Delta\mathbf{x}_{c_{\text{tool}}}(t) \approx \Delta\mathbf{x}_{c_{\text{tool}}}(t - t_c) + \mathbf{R}(\mathbf{x}_{c_{\text{normal}}})\Delta\mathbf{x}_{c_{\text{abs}}}(t) \tag{8.50}$$

where $\Delta\mathbf{x}_{c_{\text{abs}}}(t)$ is defined as

$$\Delta\mathbf{x}_{c_{\text{abs}}}(t) = \mathbf{x}_{c_{\text{abs}}}(t) - \mathbf{x}_{c_{\text{abs}}}(t - t_c) \tag{8.51}$$

Initiation of contact with the unknown surface is accomplished by moving within close proximity of the object (about 15mm), and then moving slowly (5mm per second) toward the surface. The controller monitors the force sensor for a particular force threshold (1 pound) as the end–effector's motion proceeds. When this threshold is broken the hybrid control scheme is initiated along with the desired trajectory specified along the unknown surface.

8.6 HLL Implementation Results

The integrated learning hybrid control scheme presented in the previous sections was implemented on the IBM 7545 for performance evaluation. For comparison purposes a PID

implementation of the same control topology as described in section 8.5.1 was also tested.

The IBM 7545 was required to track an unknown, irregular, curvy surface with a constant contact force of 5 pounds. The desired position trajectory was a straight line at a constant height in terms of the surface contour. This trajectory included a ramp up to a constant speed of 15mm per second using a 4^{th} order polynomial, a ramp back down to zero, and then the reverse of this motion. An entire cycle took approximately 11 seconds, and this trajectory was repeated 4 times until the repetitive controller converged to steady state operation.

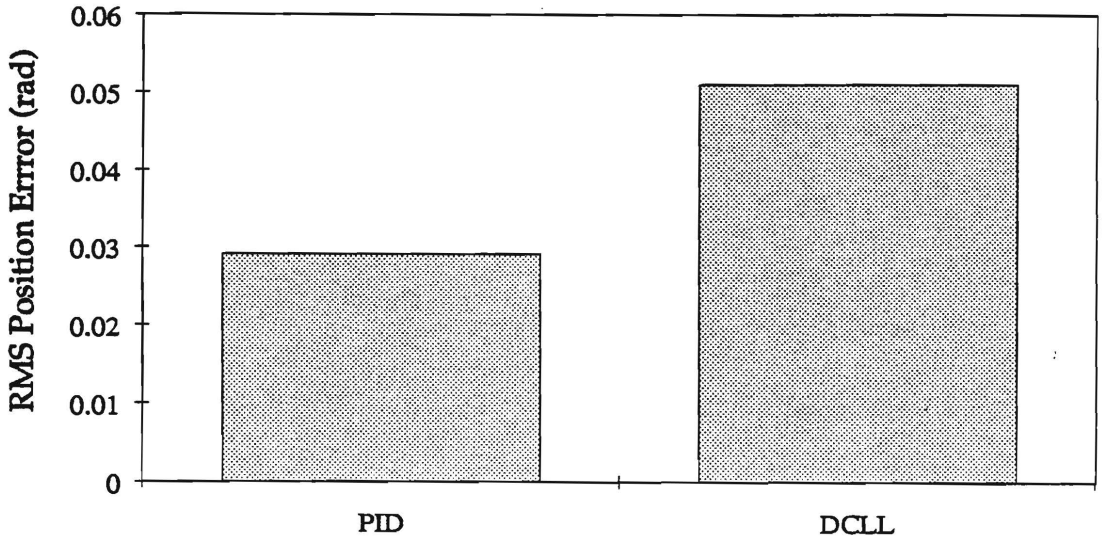

Figure 8.2: First Cycle RMS Position Error

Figure 8.2 shows the RMS position error for the first cycle, and figure 8.3 shows the force error for this same cycle for both the PID and DCLL control schemes. The PID controller outperformed the DCLL implementation by approximately 41% in terms of position error, and approximately 30% in terms of force error during this first cycle. However, this is to be expected of a repetitive type control law. Figures 8.4 and 8.5 show the RMS of the position and force error during a steady state (i.e. the fourth) cycle of the trajectory. By this time the DCLL is bettering the position error of the PID algorithm by 79%, and the RMS force error is down to 0.11 pounds (57% better than the PID).

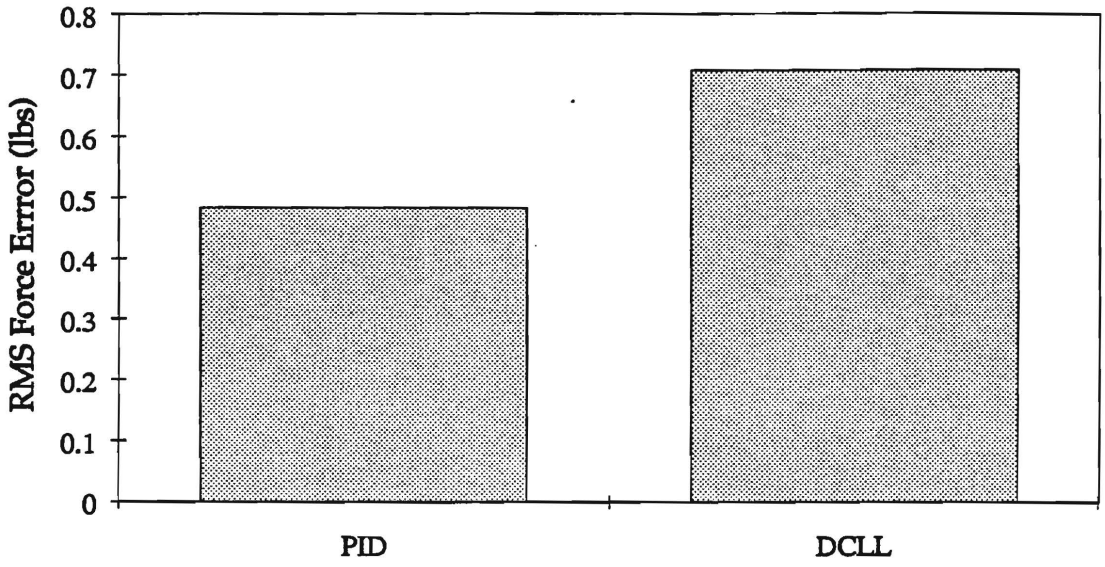

Figure 8.3: First Cycle RMS Force Error

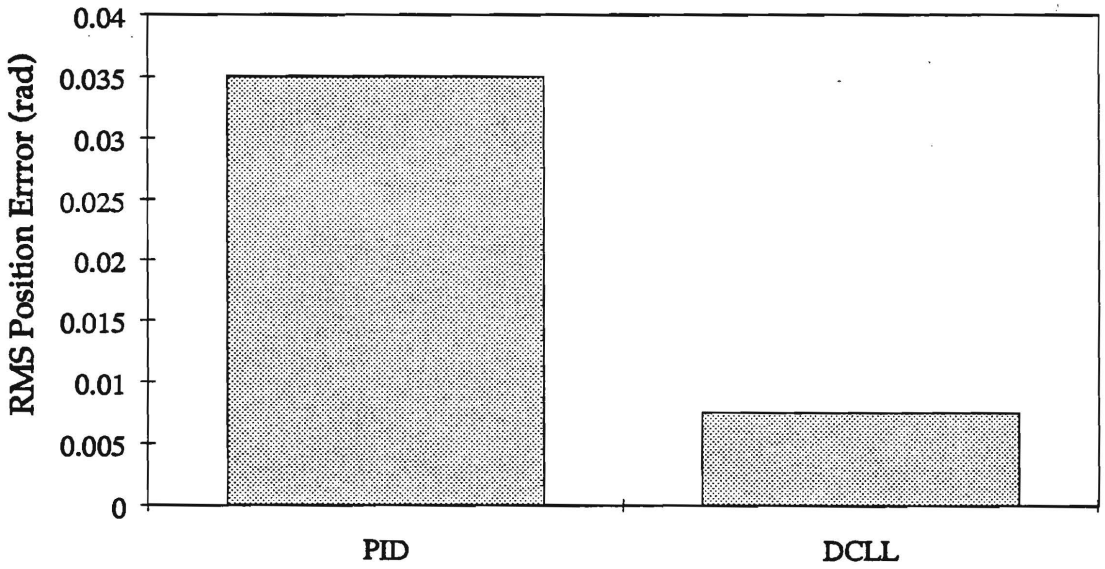

Figure 8.4: Steady State Cycle RMS Position Error

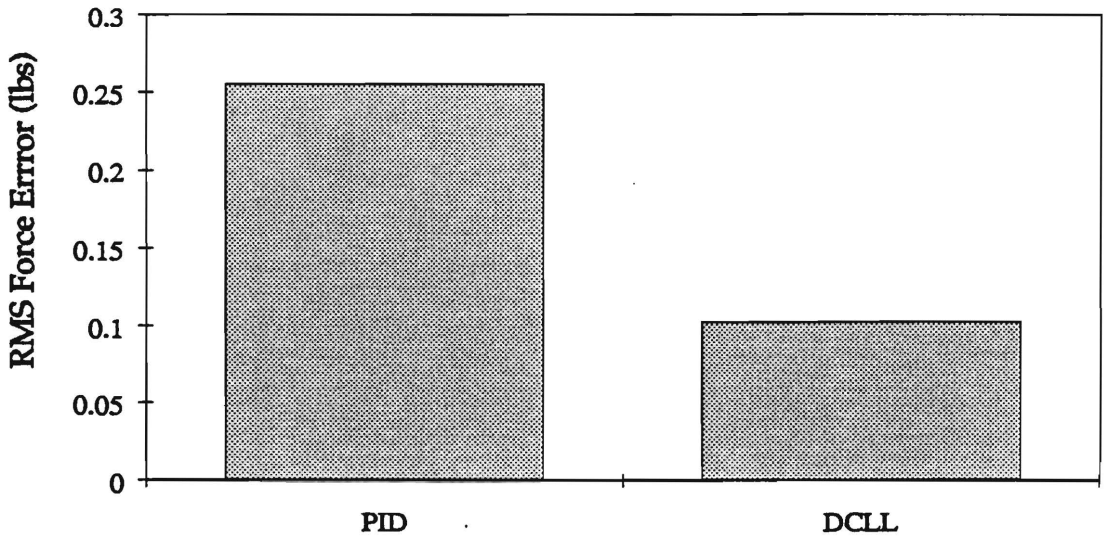

Figure 8.5: Steady State Cycle RMS Force Error

Figure 8.6 displays the "learned" contour of the unknown surface in terms of end–effector orientation. Notice the "flat spot" which occurs approximately 1/4 and 3/4 of the way through the trajectory. This represents a single flat area which is passed over during both the forward and backward motion of the manipulator. The flat areas at the trajectory end points and center point represent a constant orientation region in time while the manipulator comes to rest and reverses itself. Tracking the contact surface required a total end–effector rotation of approximately 20 degrees, and included negotiating the large flat area as well as several small ripples. Figure 8.7 shows the sensed torque at the contact point generated due to incorrect orientation for both control schemes. The DCLL does a 73% better job of estimating the correct rotation in terms of this measure.

The DCLL easily outperformed a well tuned PID implementation of the hybrid algorithm with trajectory orientation learning. The repetitive control law performed only slightly worse than the PID system during the first cycle. This shows that if execution of the task changes due to repositioning of the unknown surface, a different payload, or some other alteration, the DCLL implementation of the HLL will only suffer slightly during the first cycle after the introduction of the changed environment. Within 3 or 4 cycles the DCLL

will once again be tracking with excellent accuracy.

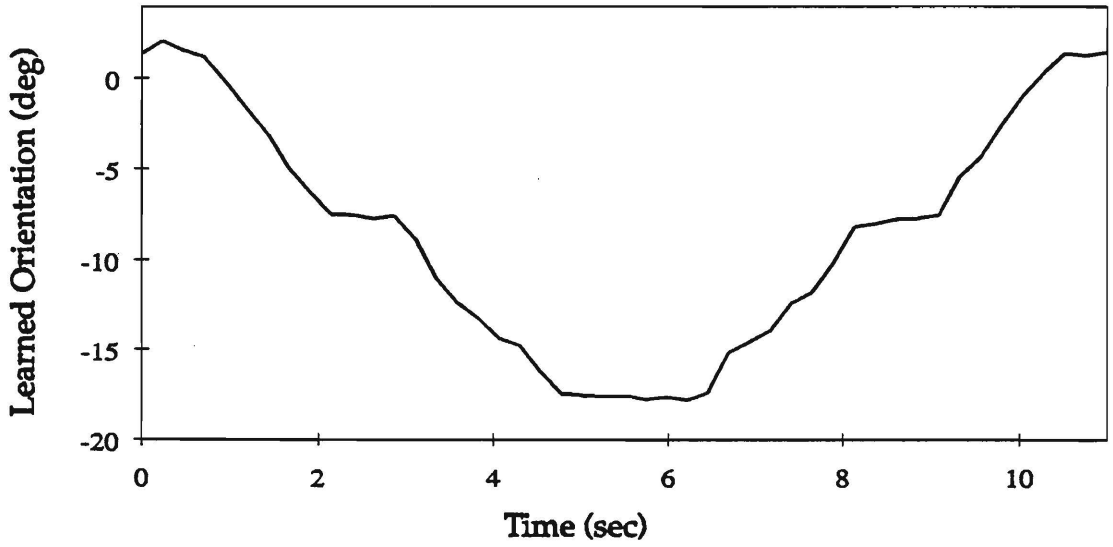

Figure 8.6: Steady State Orientation Trajectory for the DCLL

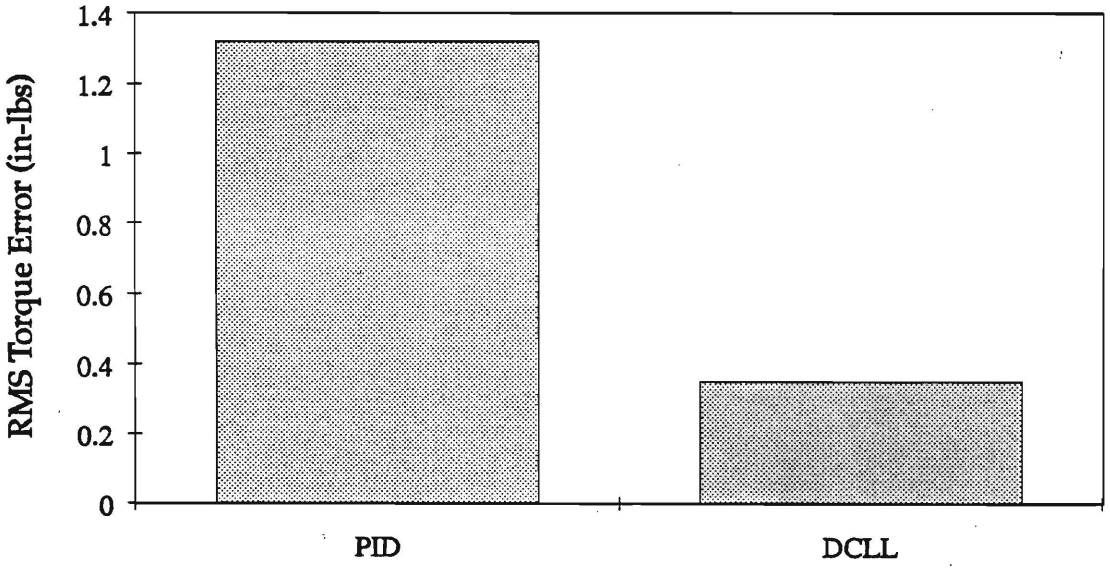

Figure 8.7: Steady State Cycle RMS Contact Generated Torque

Chapter 9

Conclusion

Several new learning control algorithms for both position and force control of robotic manipulators have been presented. All of these algorithms, as well as some previously developed schemes used for comparison purposes, were implemented on an actual IBM 7545 robot.

A joint space and Cartesian space *repetitive learning position controller* referred to as the Desired Compensation Learning Law (DCLL) formed a foundation for extension to the new algorithms. This controller was applicable to control of a robot performing a periodic task, and required no specific description of the dynamic structure of the system. Subsequent stability results for both forms of the algorithm (joint and Cartesian space) were presented. This controller was capable of the highest performance of any of the schemes examined in this dissertation. However, valid tasks were restricted to the set of periodic trajectories.

In the interest of applying learning position control to *non–periodic* tasks, a new adaptive/learning algorithm referred to as the Friction Compensation Adaptive Law (FCAL) was developed. This joint space scheme used an adaptive feedforward signal based on the manipulator's dynamic equations of motion coupled with a general learning signal capable of estimating and canceling linear and nonlinear friction effects. The friction learning term did not require modeling equations of the friction mechanism. Stability results based on the Lyapunov approach were presented, and robustness qualities were explored. This controller exhibited excellent performance in comparison to the "pure" adaptive form, and achieved

tracking accuracy almost as good as the DCLL.

Another new position controller referred to as the Configuration Learning Law (CLL) was developed in an attempt to achieve performance at least as good as the FCAL without having to specify the exact equations of motion for the system. This algorithm used the *general* form of the robot dynamic equations of motion coupled with the friction learning term of the FCAL to generate its feedforward signal. The CLL was applicable to non–periodic tasks, and resembled a neural network in its structure. However, it was computationally and memory efficient, and was easily implemented for real–time control of the IBM 7545. Stability results were presented, and the performance of the algorithm was similar to that of the FCAL.

A new *repetitive learning force controller* based on the Cartesian space DCLL scheme was also developed. This algorithm, referred to as the Hybrid Learning Law (HLL), was actually a combination of force, Cartesian position, and trajectory orientation learning control. By using a wrist mounted force sensor, the IBM 7545 was able to track an unknown surface with a specified normal contact force and tangential position trajectory. This controller did not require a specific model of the surface interaction stiffness or dynamics of the manipulator. Stability results were presented, and excellent performance was observed compared to a PID implementation of the same hybrid force/position control topology.

The work in this dissertation expands the current applications and level of development of learning control for nonlinear mechanical systems. Excellent performance results were achieved with all of the algorithms while requiring very little knowledge of the system. Future work worthy of pursuit might include non–repetitive learning force control and even greater relaxation of the structure required for formulation of non–repetitive learning position control. Also, as more computational power becomes available and feasible in an industrial environment, true neural network real–time control similar in structure to the CLL may yield ultimate performance for many nonlinear systems.

References

[1] Anderson, B. D. O. and R. M. Johnston, "Exponential Convergence of Adaptive Algorithm Identification and Control Algorithms," *Automatica,* vol. 18, pp. 1-13, 1982.

[2] Arimoto, S., S. Kawamura, and F. Miyazaki, "Bettering Operation of Robots by Learning," *Journal of Robotic Systems,* vol. 1 & 2, pp. 123-140, 1984.

[3] Asada, H. and J.–J. E. Slotine, *Robot Analysis and Control,* John Wiley & Sons, 1986.

[4] Atkeson, C. G. and J. McIntyre, "Robot Trajectory Learning Through Practice," *Proceedings of the IEEE Conference on Robotics and Automation,* San Francisco, California, 1986.

[5] Bondi, P., C. Giuseppe, and L. Gambardella, "Robust Controller Designs for Robot Manipulator Systems," *IEEE Journal of Robotics and Automation,* vol. A-4, no. 1, Feb. 1988.

[6] Bodson, M. and S. Sastry, *Adaptive Control Stability, Convergence, and Robustness,* Prentice Hall Advanced Reference Series, 1989.

[7] Boothby, W., *An Introduction to Differentiable Manifolds and Rimanian Geometry,* Academic Press, 1975.

[8] Brogan, L., *Modern Control Theory,* 3rd edition, Prentice–Hall, 1991.

[9] Chida, Y., Y. Kaku, and T. Mita, "On the Force/Trajectory Control of Robot Arms," *Advanced Robotics,* 1988, Vol. 2, p. 369.

[10] Craig, J. J., *An Introduction to Robotics,* Addison–Wesley, 1988.

[11] Craig, J. J., P. Hsu, and S. S. Sastry, "Adaptive Control of Mechanical Manipulators," *Proceedings of the 1986 IEEE International Conference on Robotics and Automation,* 1986.

[12] Guglielmo, K., and N. Sadegh, "A New Adaptive Controller with Generalized Friction Estimation," *Proceedings of the 1991 ASME Winter Annual Meeting,* Atlanta, Georgia, 1991.

[13] Guglielmo, K. and N. Sadegh, "Experimental Evaluation of a New Robot Learning Controller," *IEEE Conference on Robotics and Automation,* Sacramento, California, 1991.

[14] Guglielmo, K. and N. Sadegh, "Theory and Implementation of a Repetitive Robot Controller with Cartesian Trajectory Description," Submitted for publication to the *ASME Journal of Dynamic Systems Measurement and Control*, 1990.

[15] Guglielmo, K. and N. Sadegh, "A New Learning Controller for Mechanical Manipulators Applied in Cartesian Space," *Proceeding of the 1990 ASME Winter Annual Meeting,* Dallas, Texas, 1990.

[16] Guglielmo, K. H., "A New Learning Controller for Mechanical Manipulators Applied in Cartesian Space," Master's Thesis, Department of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia, 1990.

[17] Hara, S. and Y. Yamamoto, "Stability of Repetitive Control Systems," *Proceedings of the 24th Conference on Decision and Control,* Fort Lauderdale, Florida, Dec. 1985.

[18] Horowitz, R., W.–W. Kao, and N. Sadegh, "Robot Analysis and Control," *Proceedings of the 1989 IEEE International Conference on Robotics and Automation,* Scottsdale, Arizona, May 1989.

[19] Kang, C. G., W.–W. Kao, M. Boals, and R. Horowitz, "Modeling, Identification and Simulation of a Two Link SCARA Manipulator," *Proceedings of the ASME 1988 Winter Annual Meeting,* Chicago, Illinois, Nov. 1988.

[20] Kawamura, S., F. Miyazaki, and S. Arimoto, "Realization of Robot Motion Based on a Learning Method," *IEEE Transactions System, Man, Machine Cybernetics,* Jan. 1988.

[21] Koditschek, D., "Adaptive Techniques for Mechanical Manipulators," *Proceedings of the 5th Yale Workshop on Applications of Adaptive Systems Theory,* Yale University, May 1987.

[22] Koivo, A. J., "Adaptive Force–Position–Velocity Control for Robotic Manipulators," *Proceedings of the 23rd Annual Allerton Conference on Communication, Control, and Computing,* 1985, p. 421.

[23] Liu, L., Y. Han, R. Lingarkar, N. K. Sinha, and M. A. Elbestawi, "On Adaptive Force/Motion Control of Constrained Robots," *IEEE IECOM,* 1989, p. 433.

[24] Liu, M. H., W. S. Chang, and L. Q. Zhang, "Dynamic and Adaptive Force Controllers for Robotic Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation,* 1988, Vol. 3, p. 1478.

[25] Miller, T., R. Hewes, F. Glanz, and G. Kraft, "Real–Time Dynamic Control of an Industrial Manipulator Using a Neural–Network–Based Learning Controller," *IEEE Transactions on Robotics and Automation,* Feb. 1990, pp. 1–9.

[26] Miller, T., "Sensor Based Control of Robotic Manipulators Using a General Learning Algorithm," *IEEE Journal of Robotics and Automation,* 1987, Vol. 3, p. 157.

[27] Narendra, K. S. and L. Valavani, "A Comparison of Lyapunov and Hyperstability Approaches to Adaptive Control of Continuous Systems," *IEEE Transactions on Automatic Control,* vol. AC-25, no. 2, pp. 243-247, 1980.

[28] Royden, H. L., *Real Analysis,* McMillan Publishing, 1968.

[29] Sadegh, N., "Adaptive Control of Mechanical Manipulators: Stability and Robustness Analysis," PhD Dissertation, Department of Mechanical Engineering, University of California at Berkeley, 1987.

[30] Sadegh, N. and R. Horowitz, "Stability and Robustness Analysis of a Class of Adaptive Controllers for Robotic Manipulators," *International Journal of Robotics Research,* June 1990.

[31] Sadegh, N., and R. Horowitz, "Stability Analysis of an Adaptive Controller for Robotic Manipulators," *Proceedings of the IEEE Conference on Robotics and Automation,* 1987, Vol. 3, p. 1223.

[32] Sadegh, N., R. Horowitz, W.-W. Kao, and M. Tomizuka, "A Unified Approach to the Design of Adaptive and Repetitive Controllers for Robotic Manipulators," *Proceedings of the USA–Japan Symposium on Flexible Automation,* Minneapolis, Minnesota, July 1988.

[33] Sadegh, N. and K. Guglielmo, "A New Learning Controller for Mechanical Manipulators," *Journal of Robotic Systems,* 1990.

[34] Slotine, J.–J. E. and W. Li, "Indirect Adaptive Robot Control," *Proceedings of the IEEE International Conference on Robotics and Automation,* pp. 704-709, Philadelphia, Pennsylvania, April 1988.

[35] Slotine, J.–J. E. and W. Li, "On the Adaptive Control of Robot Manipulators," *Robots: Theory and Applications, Proceedings of the ASME Winter Annual Meeting,* Dec. 1986.

[36] Togai, M. and O. Yamano, "Learning Control and its Optimality: Analysis and Its Application to Industrial Robots," *Proceedings of the IEEE Conference on Robotics and Automation,* San Francisco, California, 1986.

[37] Tomizuka, M., T.–S. Tsao, and K.–K. Chew, "Analysis and Synthesis of Discrete Time Repetitve Controllers," *Journal of Dynamic Systems Measurement and Control,* Transactions of the ASME, Vol. 11, Num. 3, pp. 353–358, 1989.

[38] Vidyasagar, M., *Nonlinear Systems Analysis,* Prentice–Hall, 1978.